

# Computer Science/Engr

---

## Introduction To Technology

By  
Dr Jeff Drobman

# Index

## INTRO

- ❖ **Data** Codes → slide 5
- ❖ **Hardware** Models → slide 22
- ❖ CPU Performance → slide 38
- ❖ Parallelism/Other HW → slide 41
- ❖ **Software** Models → slide 47
- ❖ Top-down A/D → slide 53
- ❖ **Program** Models → slide 58
- ❖ Software *Engr* → slide 62
- ❖ **SDLC** → slide 66
- ❖ Debugging & Testing → slide 75
- ❖ Project Mgt → slide 82

### PART 1 MODELS

- ❖ HLL → slide 87
- ❖ *Hello World* → slide 98
- ❖ C++ vs Java → slide 103
- ❖ *Java* → slide 120
- ❖ Platforms → slide 129
- ❖ Tools: SDK/IDE → slide 134

### PART 2 JAVA & HLL'S

- ❖ **Algorithms** → slide 147
- ❖ **Cryptography** → slide 174
- ❖ **Theory** → slide 180

### PART 3 THEORY

- ❖ Code Structure → slide 183
- ❖ Others: C, VB → slide 194
- ❖ Design Patterns → slide 208
- ❖ Web Apps → slide 211
- ❖ Assembly Level → slide 229

### PART 4 OTHER



# Computer Science Sub-Fields

- ❖ Problem solving and Algorithms
- ❖ Programming (OOP)
- ❖ Software Engineering (SDLC, IPO, structured design, design patterns)
- ❖ Automata theory
- ❖ Systems programming
  - OS (shell, kernel, I/O)
  - Compiler construction
- ❖ Data
  - Database management & models (DBMS)
  - Data science & Mining
- ❖ Graphics (gaming, VR)
- ❖ AI
  - Game playing with Heuristics
  - Machine learning (Deep learning)
  - Pattern recognition (fingerprints, facial, etc.)
- ❖ Cryptography & Cybersecurity
- ❖ Simulation & Modeling
  - Queueing theory
- ❖ Digital System design (logic design)
- ❖ Computer Architecture (ISA, SIMD, caches, multi-threading)
- ❖ Numerical Analysis & Control (DNC)
- ❖ Information Technology (IT/CIT)

1<sup>st</sup> course

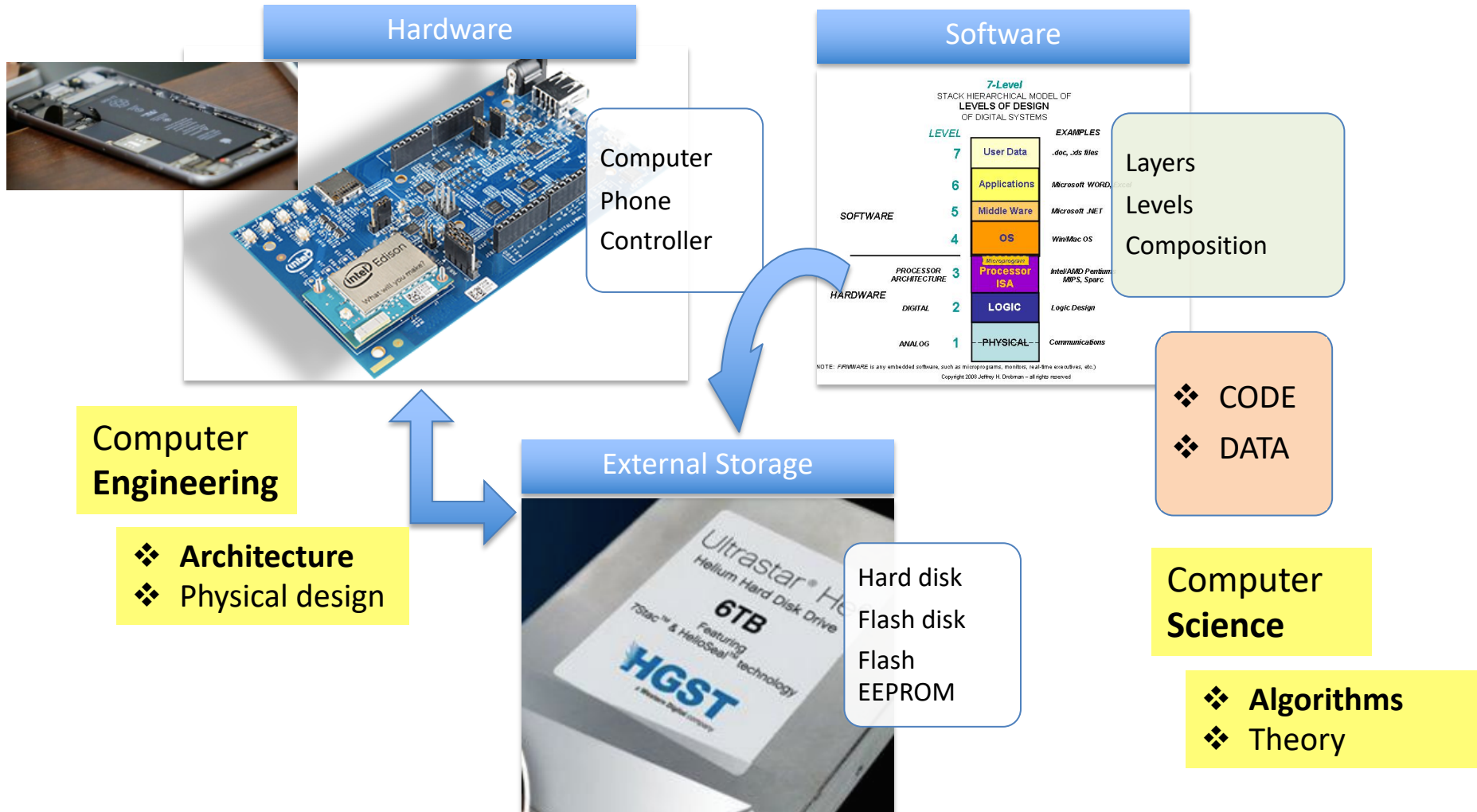
❖ Desktop

## Application Realms

- ❖ Desktop
- ❖ Mobile
- ❖ Website
- ❖ Embedded

Computer Engineering

# Digital Systems



# Software – Data

---

## Data Codes

# Ordinals

## Technical ordinals

$10^{(-24)}$  yacto  
 $10^{(-21)}$  zepto  
 $10^{(-18)}$  atto  
 $10^{(-15)}$  femto  
 $10^{(-12)}$  pico  
 $10^{(-9)}$  nano  
 $10^{(-6)}$  micro  
 $10^{(-3)}$  milli  
 $10^{(-2)}$  centi  
 $10^{(-1)}$  deci  
 $10^{(+1)}$  deka  
 $10^{(+2)}$  hecto  
 $10^{(+3)}/2^{(10)}$  kilo  
 $10^{(+6)}/2^{(20)}$  mega  
 $10^{(+9)}/2^{(30)}$  giga  
 $10^{(+12)}/2^{(40)}$  tera  
 $10^{(+15)}/2^{(50)}$  peta  
 $10^{(+18)}/2^{(60)}$  exa  
 $10^{(+21)}/2^{(70)}$  zetta  
 $10^{(+24)}/2^{(80)}$  yotta

$10^{(29)}/2^{(100)}$  geo

## Gazillions

$10^{(+6)}$  million  
 $10^{(+9)}$  billion  
 $10^{(+12)}$  trillion  
 $10^{(+15)}$  quadrillion  
 $10^{(+18)}$  quintillion  
 $10^{(+21)}$  sexillion  
 $10^{(+24)}$  septillion  
 $10^{(+27)}$  octillion  
 $10^{(+30)}$  nonillion  
 $10^{(+33)}$  decillion  
 $10^{(+36)}$  undecillion  
 $10^{(+39)}$  duodecillion  
 $10^{(+42)}$  tredecillion  
 $10^{(+45)}$  quattuordecillion  
 $10^{(+48)}$  quindecillion  
 $10^{(+51)}$  sexdecillion  
 $10^{(+54)}$  septendecillion  
 $10^{(+57)}$  octodecillion  
 $10^{(+60)}$  novemdecillion  
 $10^{(+63)}$  vigintillion  
 $10^{(+100)}$  googol  
 $10^{(+303)}$  centillion  
 $10^{(10^{(+100)})}$   
 googolplex

Ordinal	Power of 2	Power of 10	Actual
1K	$2^{10}$	$10^3$	1024
1M	$2^{20}$	$10^6$	1,048,576
1G	$2^{30}$	$10^9$	$1.074 \times 10^9$
1T	$2^{40}$	$10^{12}$	$1.0995 \times 10^{12}$

Name	$2^n$	M/G	Actual
byte	$2^8$	--	256
short	$2^{16}$	64K	65,536
word	$2^{32}$	4B	$4.3 \times 10^9$
long	$2^{64}$	16 Q	$1.84 \times 10^{19}$
IPv6	$2^{128}$	340 uD	$3.4 \times 10^{38}$

# Number Codes

---

## ❖ Invented/Artificial

- ❑ Signaling
  - Smoke signals
  - Drums
  - Semaphores
- ❑ Communications
  - Morse code
  - Hollerith code (punch cards)
  - Paper tape codes
  - Encryption/cypher codes
  - **ASCII code** (also **EBCDIC**)

## ❖ Natural

- ❑ DNA – Genetic code
  - Base-4 {A,C,G,T}
- ❑ Fibonacci sequence
  - Shell growth
  - Leaf growth

# Telegraph: Morse Code

## INTRO

Base 2 = {dot, dash}

Each letter is a 1 to 4-bit character

1<sup>st</sup> Digital Code

1836-1844

by Samuel F.B. Morse et al.

### International Morse Code

1. The length of a dot is one unit.
2. A dash is three units.
3. The space between parts of the same letter is one unit.
4. The space between letters is three units.
5. The space between words is seven units.

A	• —	U	• • —
B	— • • •	V	• • • —
C	— • — •	W	• — —
D	— • •	X	— • • —
E	•	Y	— • — —
F	• • — •	Z	— — • •
G	— — •		
H	• • • •		
I	• •		
J	• — — —		
K	— • — —		
L	• — • •		
M	— —		
N	— •		
O	— — —		
P	• — — •		
Q	— — • —		
R	• — • •		
S	• • •		
T	—		

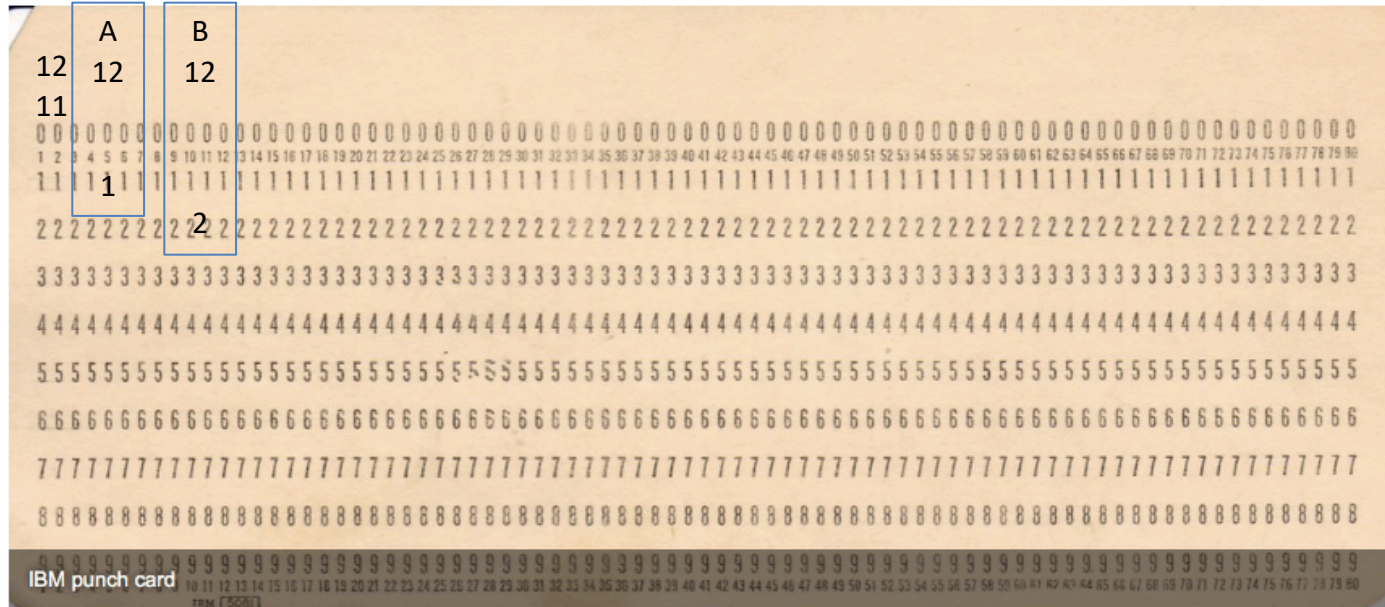
1	• — — —
2	• • — — —
3	• • • — —
4	• • • • —
5	• • • • •
6	— • • • •
7	— — • • •
8	— — — • •
9	— — — — •
0	— — — — —



A typical "straight key". This U.S. model, known as the J-38, was manufactured in huge quantities during World War II, and remains in widespread use today. In a straight key, the signal is "on" when the knob is pressed, and "off" when it is released. Length and timing of the dots and dashes are entirely controlled by the telegraphist.



# Punchcards



Invented by Herman Hollerith for 1890 census

# ASCII Codes- Letters

**Table 1-3 ASCII Conversion Chart for Letters**

Hex	Character	Hex	Character
41	A	61	a
42	B	62	b
43	C	63	c
44	D	64	d
45	E	65	e
46	F	66	f
47	G	67	g
48	H	68	h
49	I	69	i
4a	J	6a	j
4b	K	6b	k
4c	L	6c	l
4d	M	6d	m
4e	N	6e	n
4f	O	6f	o
50	P	70	p

1963



# ASCII Codes- 7-bit

USASCII code chart

<div> <div> b<sub>7</sub> b<sub>6</sub> b<sub>5</sub> </div> <div> b<sub>4</sub> b<sub>3</sub> b<sub>2</sub> b<sub>1</sub> </div> <div> Row </div> </div>					0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
					0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	\$	4	D	T	d	t
0	1	0	1	5	ENO	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	
1	1	0	1	13	CR	GS	-	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	_	o	DEL

\n=\u000A  
sp=\u0020

char ch=0xA  
char sp=0x20

# IBM EBCDIC

## EBCDIC

Extended **Binary Coded Decimal**

7-bit code used by large computers  
The collating sequence for the two codes is shown as follows

ASCII Code			EBCDIC Code		
Character	Decimal	Hex	Character	Decimal	Hex
space	32	20	space	64	40
!	33	21	.	75	4B
"	34	22	<	76	4C
#	35	23	(	77	4D
\$	36	24	+	78	4E
%	37	25	!	79	4F
&	38	26	&	80	50
single quote	39	27	\$	91	5B
(	40	28	*	92	5C
)	41	29	)	93	5D
*	42	2A	;	94	5E
+	43	2B	minus -	96	60
comma	44	2C	/	97	61
-	45	2D	comma	107	6B
.	46	2E	%	108	6C
/	47	2F	>	110	6E
0	48	30	?	111	6F
1	49	31	:	122	7A
2	50	32	#	123	7B
3	51	33	@	124	7C
4	52	34	single quote	125	7D
5	53	35	=	126	7E
6	54	36	"	127	7F
7	55	37	a	129	81
8	56	38	b	130	82
9	57	39	c	131	83
:	58	3A	d	132	84
;	59	3B	e	133	85
<	60	3C	f	134	86
=	61	3D	g	135	87
>	62	3E	h	136	88
?	63	3F	i	137	89
@	64	40	j	138	8A
A	65	41	k	139	8B
B	66	42	l	140	8C
C	67	43	m	141	8D
D	68	44	n	142	8E
E	69	45	o	143	8F
F	70	46	p	144	90
G	71	47	q	145	91
H	72	48	r	146	92
I	73	49	s	147	93
J	74	4A	t	148	94
K	75	4B	u	149	95
L	76	4C	v	150	96
M	77	4D	w	151	97
N	78	4E	x	152	98
O	79	4F	y	153	99
P	80	50	z	154	9A
Q	81	51	A	169	A9
R	82	52	B	170	AA
S	83	53	C	171	AB
T	84	54	D	172	AC
U	85	55	E	173	AD
V	86	56	F	174	AE
W	87	57	0	193	C1
X	88	58	1	194	C2
Y	89	59	2	195	C3
Z	90	5A	3	196	C4
a	97	61	4	197	C5
b	98	62	5	198	C6
c	99	63	6	199	C7
d	100	64	7	200	C8
e	101	65	8	201	C9
f	102	66	9	202	CA
g	103	67		203	CB
h	104	68		204	CC
i	105	69		205	CD
j	106	6A		206	CE
k	107	6B		207	CF
l	108	6C		208	D0
m	109	6D		209	D1
n	110	6E		210	D2
o	111	6F		211	D3
p	112	70		212	D4
q	113	71		213	D5
r	114	72		214	D6
s	115	73		215	D7
t	116	74		216	D8
u	117	75		217	D9
v	118	76		218	DA
w	119	77		219	DB
x	120	78		220	DC
y	121	79		221	DD
z	122	7A		222	DE
				223	DF
				224	E0
				225	E1
				226	E2
				227	E3
				228	E4
				229	E5
				230	E6
				231	E7
				232	E8
				233	E9
				234	EA
				235	EB
				236	EC
				237	ED
				238	EE
				239	EF
				240	F0
				241	F1
				242	F2
				243	F3
				244	F4
				245	F5
				246	F6
				247	F7
				248	F8
				249	F9

# Old Mac Char Codes

16-bit

Second digit	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	DLE	space	0	@	P	`	p	Ä	ê	†	∞	¿	-		
1	SOH	DC1	!	1	A	Q	a	q	Å	ë	•	±	ı	—		
2	STX	DC2	"	2	B	R	b	r	Ç	í	‡	≤	¬	“		
3	ETX	DC3	#	3	C	S	c	s	É	ì	£	≥	√	”		
4	EOT	DC4	\$	4	D	T	d	t	Ñ	î	§	¥	ƒ	‘		
5	ENQ	NAK	%	5	E	U	e	u	Ö	ï	●	μ	≈	’		
6	ACK	SYN	&	6	F	V	f	v	Ü	ñ	¶	∂	Δ	÷		
7	BEL	ETB	'	7	G	W	g	w	á	ó	ß	Σ	«	◊		
8	BS	CAN	(	8	H	X	h	x	à	ò	®	Π	»	ÿ		
9	HT	EM	)	9	I	Y	i	y	â	ô	©	π	...			
A	LF	SUB	*	:	J	Z	j	z	ä	ö	™	∫	—			
B	VT	ESC	+	;	K	[	k	{	ã	õ	’	æ	À			
C	FF	FS	,	<	L	\	l		å	ú	”	ø	Ã			
D	CR	GS	-	=	M	]	m	}	ç	ù	≠	Ω	Õ			
E	SO	RS	.	>	N	^	n	~	é	û	Æ	æ	Œ			
F	SI	US	/	?	O	_	o	DEL	è	ü	Ø	ø	œ			

unique  
special chars

— stands for a nonbreaking space, the same width as a digit.

The shaded characters cannot normally be generated from the Macintosh keyboard or keypad.

Figure 1. Macintosh Character Set



## Unicode

From Wikipedia, the free encyclopedia

*Not to be confused with [Unicode \(telegraphy\)](#).*

*For what the term "Unicode" means in Microsoft documentation, see [UTF-16](#).*

**Unicode** is a [information technology standard](#) for the consistent [encoding](#), representation, and handling of [text](#) expressed in most of the world's [writing systems](#). The standard is maintained by the [Unicode Consortium](#), and as of March 2020 the most recent version, *Unicode 13.0*, contains a repertoire of 143,924<sup>[1]</sup> [characters](#) (consisting of 143,696 graphic characters, 163 format characters and 65 control characters) covering 154 modern and historic [scripts](#), as well as multiple symbol sets and [emoji](#). The character repertoire of the Unicode Standard is synchronized with [ISO/IEC 10646](#), and both are code-for-code identical.

The *Unicode Standard* consists of a set of code charts for visual reference, an encoding method and set of standard [character encodings](#), a set of reference [data files](#), and a number of related items, such as character properties, rules for [normalization](#), decomposition, [collation](#), rendering, and [bidirectional text](#) display order (for the correct display of text containing both right-to-left scripts, such as [Arabic](#) and [Hebrew](#), and left-to-right scripts).<sup>[2]</sup>

Unicode's success at unifying character sets has led to its widespread and predominant use in the [internationalization](#) and [localization](#) of computer [software](#). The standard has been implemented in many recent technologies, including modern [operating systems](#), [XML](#), [Java](#) (and other programming languages), and the [.NET Framework](#).

Unicode can be implemented by different [character encodings](#). The Unicode standard defines [UTF-8](#), [UTF-16](#), and [UTF-32](#), and several other encodings are in use. The most commonly used encodings are UTF-8, UTF-16, and [UCS-2](#) (without full support for Unicode), a precursor of UTF-16; [GB18030](#) is standardized in China and implements Unicode fully, while not an official Unicode standard.

[UTF-8](#), the dominant encoding on the [World Wide Web](#) (used in over 94% of websites as of November 2019),<sup>[3]</sup> uses one [byte](#)<sup>[note 1]</sup> for the first 128 [code points](#), and up to 4 bytes for other characters.<sup>[4]</sup> The first 128 Unicode code points represent the [ASCII](#) characters, which means that any ASCII text is also a UTF-8 text.

[UCS-2](#) uses two bytes (16 bits) for each character but can only encode the first 65,536 code points, the so-called [Basic Multilingual Plane](#) (BMP). With 1,112,064 possible Unicode code points corresponding to characters (see [below](#)) on 17 planes, and with over 143,000 code points defined as of version 13.0, UCS-2 is only able to represent less than half of all encoded Unicode characters. Therefore, UCS-2 is outdated, though still widely used in software. UTF-16

### Unicode



Logo of the Unicode Consortium

<b>Alias(es)</b>	<a href="#">Universal Coded Character Set</a> (UCS)
<b>Language(s)</b>	International
<b>Standard</b>	Unicode Standard
<b>Encoding formats</b>	<a href="#">UTF-8</a> , <a href="#">UTF-16</a> , <a href="#">GB18030</a> <b>Less common:</b> <a href="#">UTF-32</a> , <a href="#">BOCU</a> , <a href="#">SCSU</a> , <a href="#">UTF-7</a>
<b>Preceded by</b>	<a href="#">ISO 8859</a> , various others

# Unicode – 16-Bit

UTF-16

❖ 7 LSB are same codes as for **ASCII**

❖ 9 MSB add  $2^{16}=65,536 - 128$  new codes

❖ Japanese character sets

(漢字?),

➤ *Kanji* uses same 5000 characters as base Chinese

➤ *Hiragana/Katakana* uses (ひらがな or 平仮名?) :

(カタカナ or 片仮名?).

❖ Other foreign languages

❑ alphabets

❑ special characters

(vis-à-vis, *oomlaut*)

Initial repertoire covers these scripts: Arabic, Armenian, Bengali, Bopomofo, Cyrillic, Devanagari, Georgian, Greek and Coptic, Gujarati, Gurmukhi, Hangul, Hebrew, Hiragana, Kannada, Katakana, Lao, Latin, Malayalam, Oriya, Tamil, Telugu, Thai, and Tibetan.<sup>[19]</sup>

π Я 音 æ∞

Many modern applications can render a substantial subset of the many scripts in Unicode, as demonstrated by this screenshot from the [OpenOffice.org](https://www.openoffice.org) application.

## Unicode Transformation Format and Universal Coded Character Set [edit]

Unicode defines two mapping methods: the *Unicode Transformation Format* (UTF) encodings, and the *Universal Coded Character Set* (UCS) encodings.

The Unicode codespace is divided into seventeen *planes*, numbered 0 to 16:

Unicode planes and used code point ranges [hide]					
Basic	Supplementary				
Plane 0	Plane 1	Plane 2	Planes 3–13	Plane 14	Planes 15–16
0000–FFFF	10000–1FFFF	20000–2FFFF	30000–DFFFF	E0000–EFFFF	F0000–10FFFF
Basic Multilingual Plane	Supplementary Multilingual Plane	Supplementary Ideographic Plane	unassigned	Supplementary Special-purpose Plane	Supplementary Private Use Area planes
BMP	SMP	SIP	—	SSP	SPUA-A/B

# Unicode

Upper 128 chars of 8-bit Plane 0

UTF-16

## C1 Controls and Latin-1 Supplement<sup>[1]</sup>

Official Unicode Consortium code chart  (PDF)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
U+008x	XXX	XXX	BPH	NBH	IND	NEL	SSA	ESA	HTS	HTJ	VTs	PLD	PLU	RI	SS2	SS3
U+009x	DCS	PU1	PU2	STS	CCH	MW	SPA	EPA	SOS	XXX	SCI	CSI	ST	OSC	PM	APC
U+00Ax	NB SP	ı	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	SHY	®	¯
U+00Bx	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
U+00Cx	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
U+00Dx	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
U+00Ex	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
U+00Fx	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

### Notes

1. ^ As of Unicode version 12.0

# MS Windows (1252)

<b>1</b>	!	1	A	Q	a	q	NOT USED	'	;	±	Á	Ñ	á	ñ
	33	49	65	81	97	113	129	145	161	177	193	209	225	241
<b>2</b>	"	2	B	R	b	r	,	'	ø	²	Â	Ò	â	ò
	34	50	66	82	98	114	130	146	162	178	194	210	226	242
<b>3</b>	#	3	C	S	c	s	f	“	£	³	Ã	Ó	ã	ó
	35	51	67	83	99	115	131	147	163	179	195	211	227	243
<b>4</b>	\$	4	D	T	d	t	„	”	¤	´	Ä	Ô	ä	ô
	36	52	68	84	100	116	132	148	164	180	196	212	228	244
<b>5</b>	%	5	E	U	e	u	...	•	¥	µ	Å	Õ	å	õ
	37	53	69	85	101	117	133	149	165	181	197	213	229	245
<b>6</b>	&	6	F	V	f	v	†	-	!	¶	Æ	Ö	æ	ö
	38	54	70	86	102	118	134	150	166	182	198	214	230	246
<b>7</b>	'	7	G	W	g	w	‡	-	§	·	Ç	×	ç	÷
	39	55	71	87	103	119	135	151	167	183	199	215	231	247
<b>8</b>	(	8	H	X	h	x	^	~	¨	ˆ	È	Ø	è	ø
	40	56	72	88	104	120	136	152	168	184	200	216	232	248
<b>9</b>	)	9	I	Y	i	y	%o	™	©	¹	É	Ù	é	ù
	41	57	73	89	105	121	137	153	169	185	201	217	233	249
<b>A</b>	*	:	J	Z	j	z	Š	š	Ž	ž	Ê	Ú	ê	ú
	42	58	74	90	106	122	138	154	170	186	202	218	234	250
<b>B</b>	+	;	K	[	k	{	<	>	«	»	Ë	Û	ë	û
	43	59	75	91	107	123	139	155	171	187	203	219	235	251
<b>C</b>	,	<	L	\	l		Œ	œ	¬	¼	Ì	Û	ì	ü
	44	60	76	92	108	124	140	156	172	188	204	220	236	252
<b>D</b>	-	=	M	]	m	}	NOT USED	NOT USED	SHY	½	Í	Ý	í	ý
	45	61	77	93	109	125	141	157	173	189	205	221	237	253
<b>E</b>	.	>	N	^	n	~	NOT USED	NOT USED	®	¾	Î	Þ	î	þ
	46	62	78	94	110	126	142	158	174	190	206	222	238	254
<b>F</b>	/	?	O	_	o		NOT USED	ÿ	-	;	Ï	ß	ï	ÿ
	47	63	79	95	111	127	143	159	175	191	207	223	239	255

## INTRO

## UTF-8

From Wikipedia, the free encyclopedia

**UTF-8** is a [variable width character encoding](#) capable of encoding all 1,112,064<sup>[1]</sup> valid [code points](#) in [Unicode](#) using one to four 8-bit [bytes](#).<sup>[2]</sup> The encoding is defined by the Unicode Standard, and was originally designed by [Ken Thompson](#) and [Rob Pike](#).<sup>[3][4]</sup> The name is derived from *Unicode (or Universal Coded Character Set) Transformation Format – 8-bit*.<sup>[5]</sup>

It was designed for [backward compatibility](#) with [ASCII](#). Code points with lower numerical values, which tend to occur more frequently, are encoded using fewer bytes. The first 128 characters of Unicode, which correspond one-to-one with ASCII, are encoded using a single octet with the same binary value as ASCII, so that valid ASCII text is valid UTF-8-encoded Unicode as well. Since ASCII bytes do not occur when encoding non-ASCII code points into UTF-8, UTF-8 is safe to use within most programming and document languages that interpret certain ASCII characters in a special way, such as `"/"` ([slash](#)) in filenames, `"\"` ([backslash](#)) in [escape sequences](#), and `"%"` in [printf](#).

Since 2009, UTF-8 has been the dominant encoding (of any kind, not just of Unicode encodings) for the [World Wide Web](#) (and declared mandatory "for all things" by [WHATWG](#)<sup>[7]</sup>) and as of June 2019 accounts for 93.6% of all web pages (some of which are simply [ASCII](#), as it is a subset of UTF-8) and 95% of the top 1,000 highest ranked<sup>[8]</sup> web pages. The next-most popular multi-byte encodings, [Shift JIS](#) and [GB 2312](#), have 0.4% and 0.3% respectively.<sup>[9][10][6]</sup> The [Internet Mail Consortium](#) (IMC) recommended that all e-mail programs be able to display and create mail using UTF-8,<sup>[11]</sup> and the [W3C](#) recommends UTF-8 as the *default encoding* in [XML](#) and [HTML](#).<sup>[12]</sup>

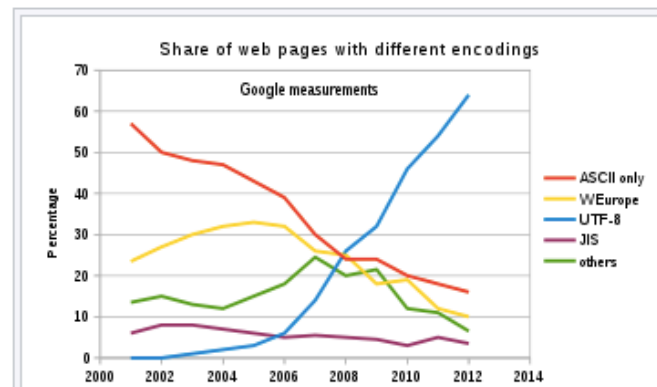
### UTF-8

<b>Language(s)</b>	International
<b>Standard</b>	Unicode Standard
<b>Classification</b>	<a href="#">Unicode Transformation Format</a> , <a href="#">extended ASCII</a> , <a href="#">variable-width encoding</a>
<b>Extends</b>	<a href="#">US-ASCII</a>
<b>Transforms /</b>	<a href="#">ISO 10646 (Unicode)</a>
<b>Encodes</b>	
<b>Preceded by</b>	<a href="#">UTF-1</a>

V · T · E

### Contents [hide]

- Description
  - Examples
  - Codepage layout
  - Overlong encodings
  - Invalid byte sequences
  - Invalid code points
- Official name and variants



Usage of the main encodings on the web from 2001 to 2012 as recorded by Google,<sup>[6]</sup> with UTF-8 overtaking all others in 2008 and over 60% of the web in 2012. Note that the ASCII-only figure includes all web pages that only contains ASCII characters, regardless of the declared header.



# UTF-8

## Variable Length version of Unicode

Number of bytes	Bits for code point	First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
1	7	U+0000	U+007F	0xxxxxxx			
2	11	U+0080	U+07FF	110xxxxx	10xxxxxx		
3	16	U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
4	21	U+10000	U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

# UTF-8 Variable Codes

## UTF-8 encoding of the ISO/IEC 10646 code points

	First	Last					
	UCS Code	Code					
Bits	Point	Point	Bytes	Byte 1	Byte 2	Byte 3	Byte 4
7	U+0000	U+007F	1	0xxxxxxx			
11	U+0080	U+07FF	2	110xxxxx	10xxxxxx		
16	U+0800	U+FFFF	3	1110xxxx	10xxxxxx	10xxxxxx	
21	U+10000	U+10FFFF	4	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

1. If the most significant bit of a byte is zero, then it is a single-byte character, and is completely ASCII-compatible.
2. If the two most significant bits in a byte are set to one, then the byte is the beginning of a multi-byte character.
3. If the most significant bit is set to one, and the second most significant bit is set to zero, then the byte is part of a multi-byte character, but is not the first byte in that sequence.

# UTF-8 (low)

## INTRO

### UTF-8

	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
0_	NUL 0000	SOH 0001	STX 0002	ETX 0003	EOT 0004	ENQ 0005	ACK 0006	BEL 0007	BS 0008	HT 0009	LF 000A	VT 000B	FF 000C	CR 000D	SO 000E	SI 000F
1_	DLE 0010	DC1 0011	DC2 0012	DC3 0013	DC4 0014	NAK 0015	SYN 0016	ETB 0017	CAN 0018	EM 0019	SUB 001A	ESC 001B	FS 001C	GS 001D	RS 001E	US 001F
2_	SP 0020	! 0021	" 0022	# 0023	\$ 0024	% 0025	& 0026	' 0027	( 0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
3_	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
4_	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
5_	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[ 005B	\ 005C	] 005D	^ 005E	_ 005F
6_	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
7_	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	 007C	} 007D	~ 007E	DEL 007F

# UTF-8 (high)

8_	• +00	• +01	• +02	• +03	• +04	• +05	• +06	• +07	• +08	• +09	• +0A	• +0B	• +0C	• +0D	• +0E	• +0F
9_	• +10	• +11	• +12	• +13	• +14	• +15	• +16	• +17	• +18	• +19	• +1A	• +1B	• +1C	• +1D	• +1E	• +1F
A_	• +20	• +21	• +22	• +23	• +24	• +25	• +26	• +27	• +28	• +29	• +2A	• +2B	• +2C	• +2D	• +2E	• +2F
B_	• +30	• +31	• +32	• +33	• +34	• +35	• +36	• +37	• +38	• +39	• +3A	• +3B	• +3C	• +3D	• +3E	• +3F
2 C_	2 0000	2 0040	LATIN 0080	LATIN 00C0	LATIN 0100	LATIN 0140	LATIN 0180	LATIN 01C0	LATIN 0200	IPA 0240	IPA 0280	IPA 02C0	ACCENTS 0300	ACCENTS 0340	GREEK 0380	GREEK 03C0
2 D_	CYRIL 0400	CYRIL 0440	CYRIL 0480	CYRIL 04C0	CYRIL 0500	ARMENI 0540	HEBREW 0580	HEBREW 05C0	ARABIC 0600	ARABIC 0640	ARABIC 0680	ARABIC 06C0	SYRIAC 0700	ARABIC 0740	THAANA 0780	N' KO 07C0
3 E_	INDIC 0800	MISC. 1000	SYMBOL 2000	KANA... 3000	CJK 4000	CJK 5000	CJK 6000	CJK 7000	CJK 8000	CJK 9000	ASIAN A000	HANGUL B000	HANGUL C000	HANGUL D000	PUA E000	FORMS F000
4 F_	SMP... 10000	□ 40000	□ 80000	SSP... C0000	SPU... 100000	4 140000	4 180000	4 1C0000	5 200000	5 1000000	5 2000000	5 3000000	6 4000000	6 40000000		

**Orange** cells with a large dot are continuation bytes. The hexadecimal number shown after a "+" plus sign is the value of the six bits they add.

**White** cells are the leading bytes for a sequence of multiple bytes, the length shown at the left edge of the row. The text shows the Unicode blocks encoded by sequences starting with this byte, and the hexadecimal code point shown in the cell is the lowest character value encoded using that leading byte.

**Red** cells must never appear in a valid UTF-8 sequence. The first two red cells (C0 and C1) could be used only for a two-byte encoding of a 7-bit ASCII character which should be encoded in one byte; as described below such "overlong" sequences are disallowed. The red cells in the F row (F5 to FD) indicate leading bytes of 4-byte or longer sequences that cannot be valid because they would encode code points larger than the U+10FFFF limit of Unicode (a limit derived from the maximum code point encodable in [UTF-16](#)), and FE and FF were never defined for any purpose in UTF-8.

**Pink** cells are the leading bytes for a sequence of multiple bytes, of which some, but not all, possible continuation sequences are valid. E0 and F0 could start overlong encodings, in this case the lowest non-overlong-encoded code point is shown. F4 can start code points greater than U+10FFFF which are invalid. ED can start the encoding of a code point in the range U+D800–U+DFFF; these are invalid since they are reserved for UTF-16 surrogate halves.

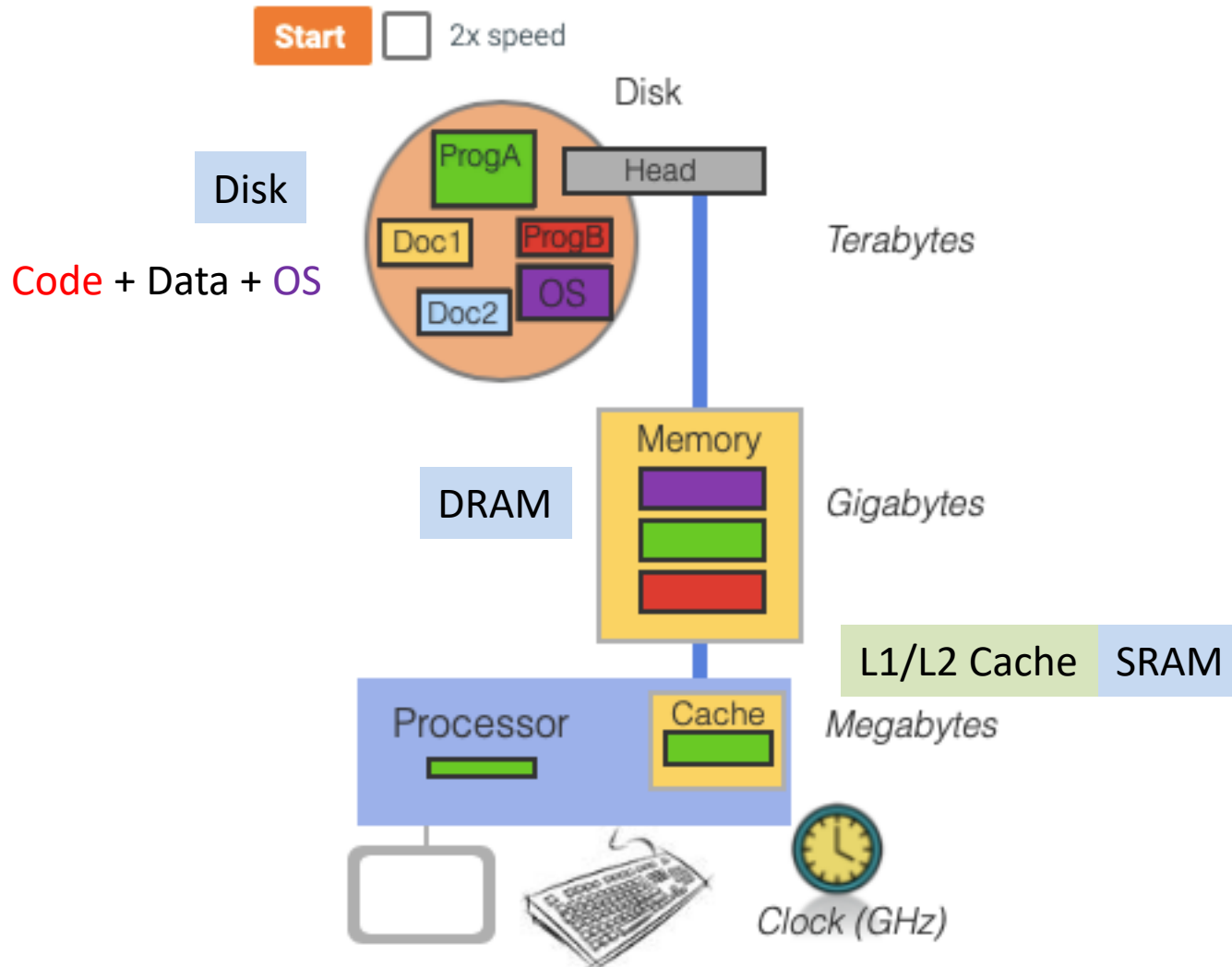
# Hardware

Computer  
Engineering

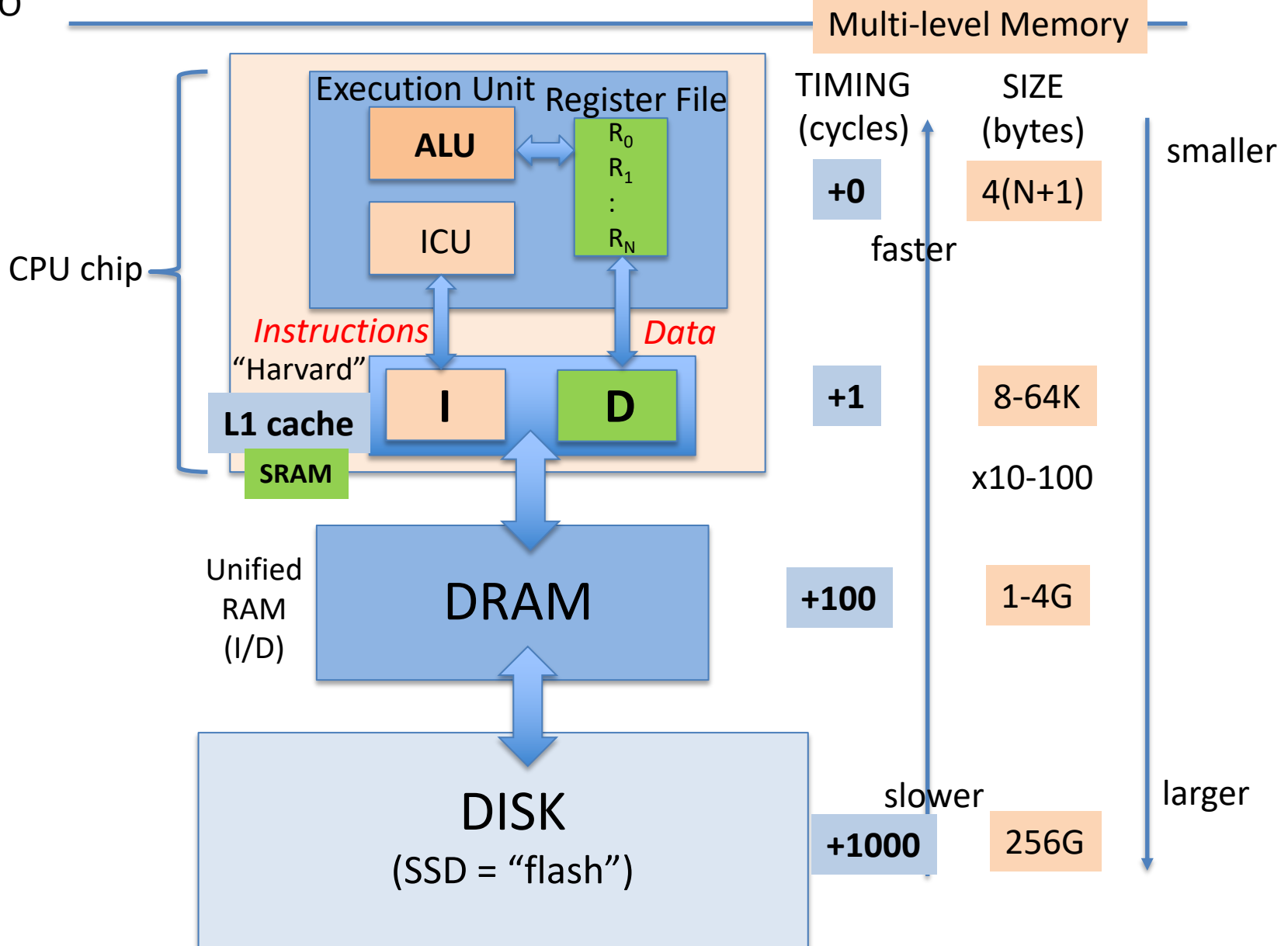
- Models
- History

# Computer Memory Org

## 1.6.1: Some computer components.



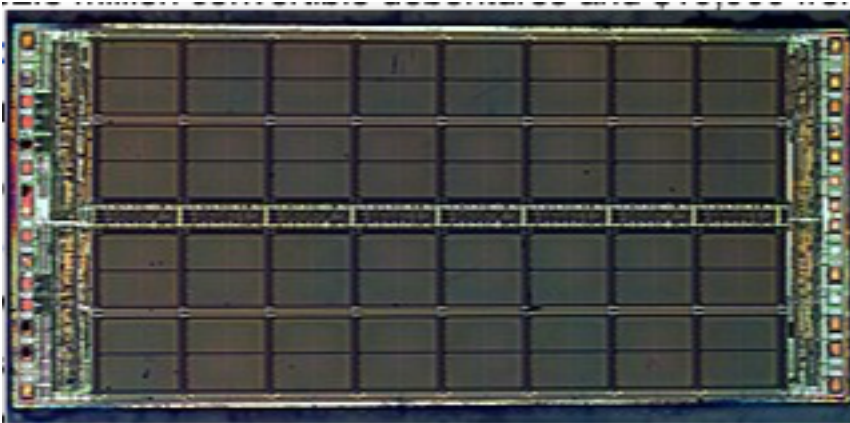
# CPU Org + Memory Hierarchy





# Memory Chips

DRAM 1T



**Dynamic random-access memory (DRAM)** is a type of random access semiconductor memory that stores each bit of data in a memory cell consisting of a tiny capacitor and a transistor, typically a MOSFET. The capacitor can either be charged or discharged; these two states are taken to

SRAM 4T

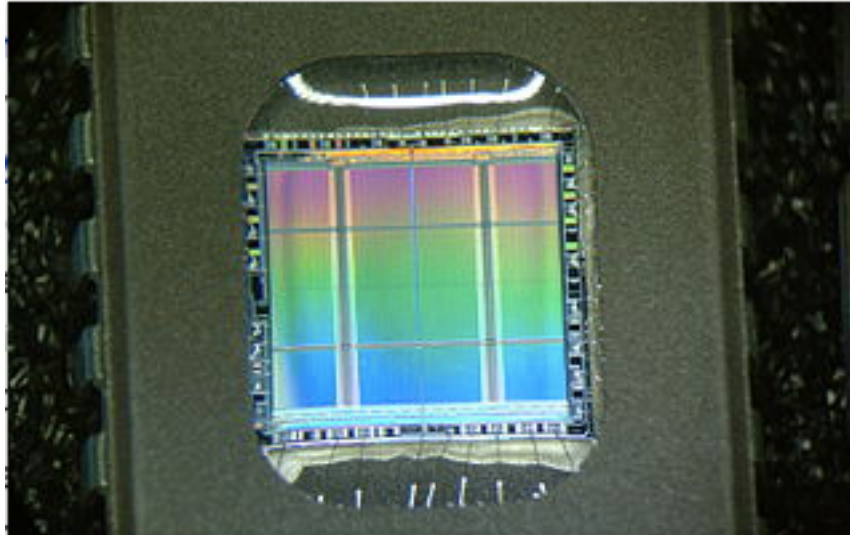


**Static random-access memory** is a type of semiconductor random-access memory (RAM) that uses bistable latching circuitry (flip-flop) to store each bit. SRAM exhibits data remanence, but it is still *volatile* in the conventional sense that data is eventually lost when the memory is not powered.



# Memory Chips

## ROM



**Read-only memory (ROM)** is a type of non-volatile memory used in computers and other electronic devices. Data stored in ROM cannot be electronically modified after the manufacture of the memory device. Read-only memory is useful for storing software that is rarely changed during the life of the system.

- ❖ ROM (masked)
- ❖ PROM
- ❖ EPROM
- ❖ EEPROM
- ❖ Flash E<sup>2</sup>

# Hardware-System Model

## DROBMAN MODEL

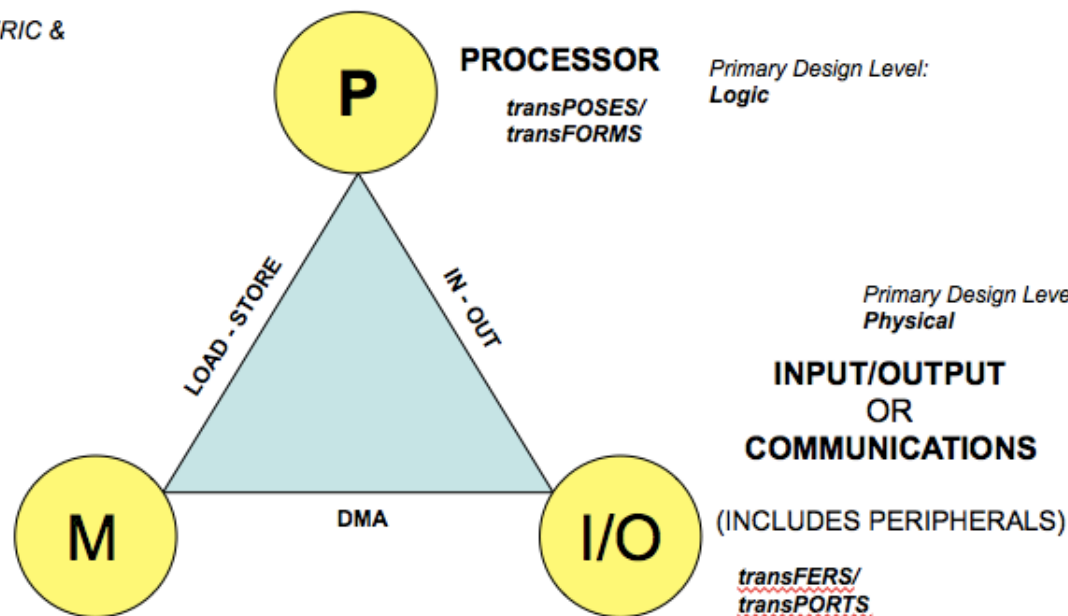
- PRIMARY FUNCTIONS ARE NODES
- NODES ALSO CONTAIN THE OTHER TWO SUBORDINATE FUNCTIONS
- NODES ARE HIERARCHICAL
- INTERCONNECTIONS ARE GENERIC & BIDIRECTIONAL
- NODES MAY BE ROTATED FOR EMPHASIS (PRIMARY SYSTEM)

Primary Design Level:  
Logic- Memory Cell

**MEMORY/  
STORAGE**

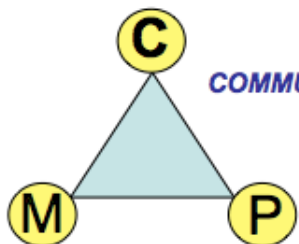
transFIXES

## TRIPARTITE GRAPH MODEL OF DIGITAL SYSTEMS: **COMPUTER**



**COMMUNICATIONS**

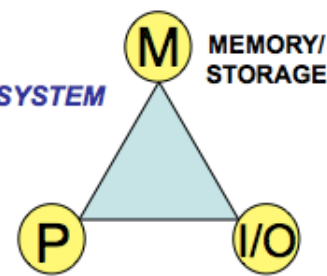
**COMMUNICATIONS SYSTEM**



**MEMORY/  
STORAGE**

**PROCESSOR**  
(E.G., NETWORK PROCESSOR)

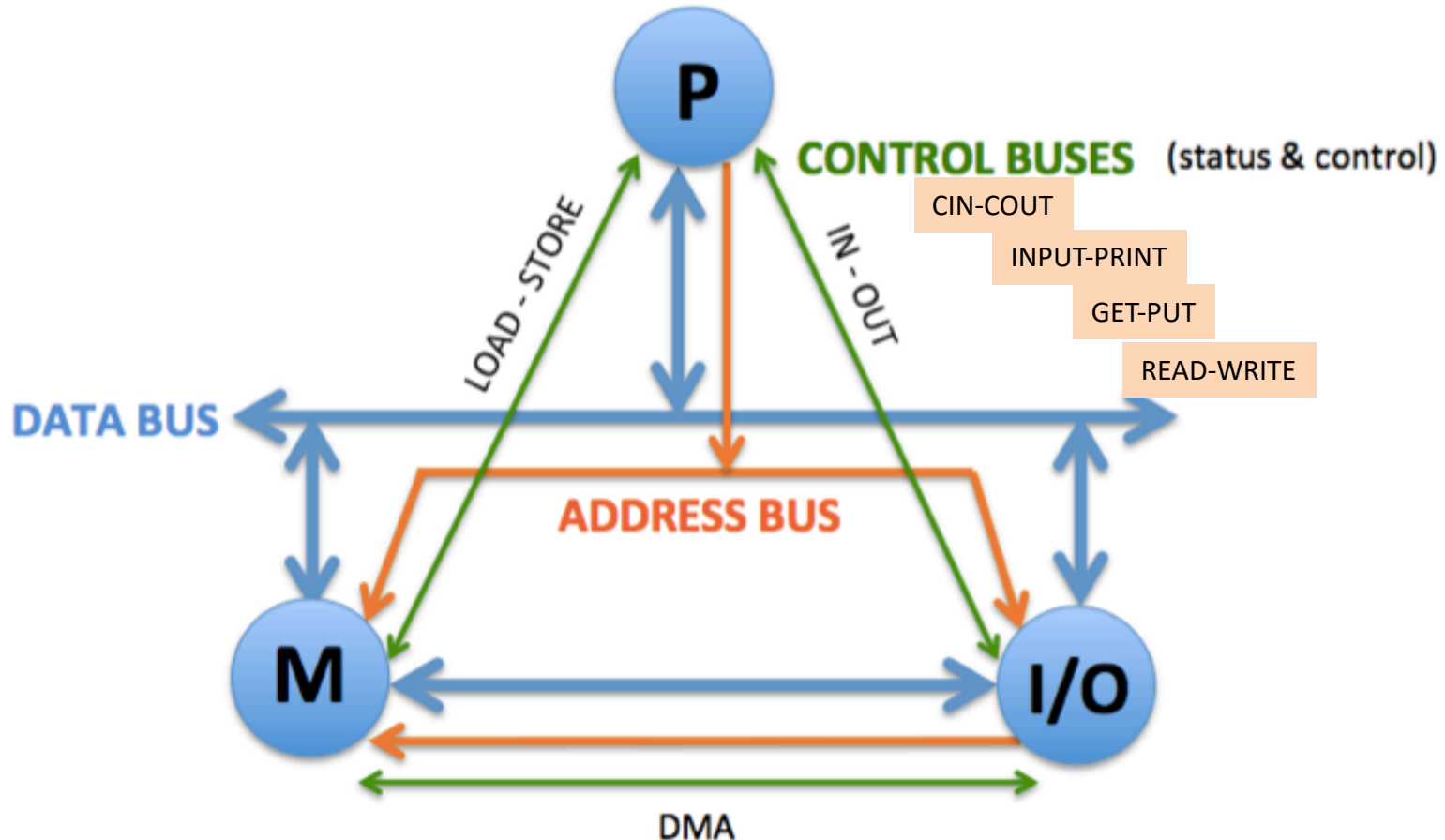
**STORAGE SYSTEM**



**PROCESSOR**

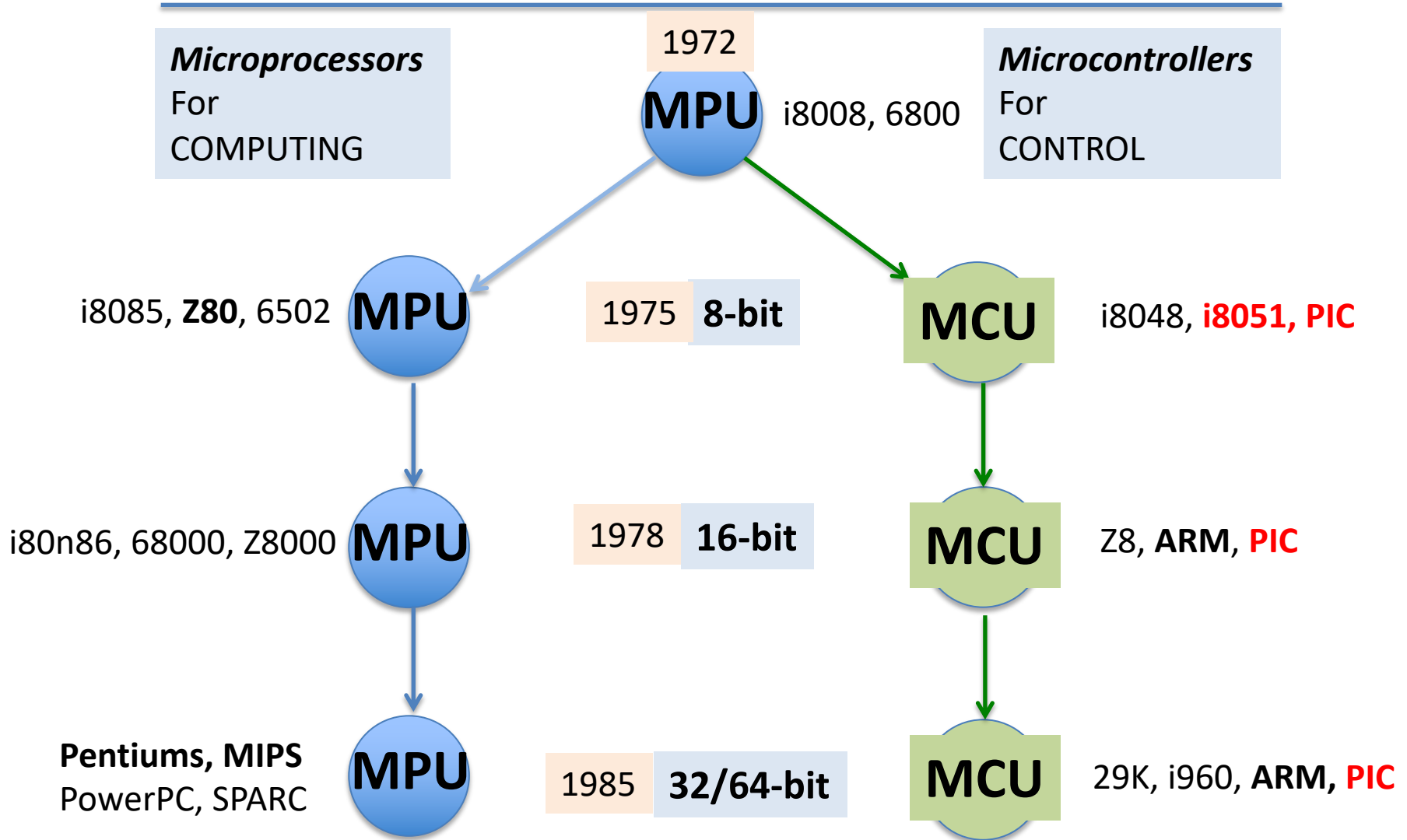
**INPUT/OUTPUT**

# Hardware-Bus Model



*NON-MULTIPLEXED BUSES*

# MPU/MCU Generations



# CISC vs RISC:

## *Complex/Reduced Instruction Set Architecture*

### ❖ Microprocessor History

- 1971-85: **CISC** (8/16-bit)
  - ✧ Intel i4004 (4-bit)
  - ✧ Intel i8008 (8-bit) → i8080 → i8085, Z80 → i8086 (16-bit) → “x86”
  - ✧ Motorola 6800 (8-bit) → 6502 → 68000 (16-bit)
  - ✧ IBM PC used i8088 (8/16-bit) in 1981 → i80n86 (“x86”) → *Pentiums*
- 1985-2000: **RISC** – (32/64-bit)
  - ✧ SPARC\* (UC Berkeley → Sun/Oracle)
  - ✧ MIPS\* (Stanford)
  - ✧ PowerPC\* (Motorola/IBM)
  - ✧ AMD 29K
  - ✧ Intel i960
  - ✧ Intel/AMD “Pentiums”\*
  - ✧ ARM\*

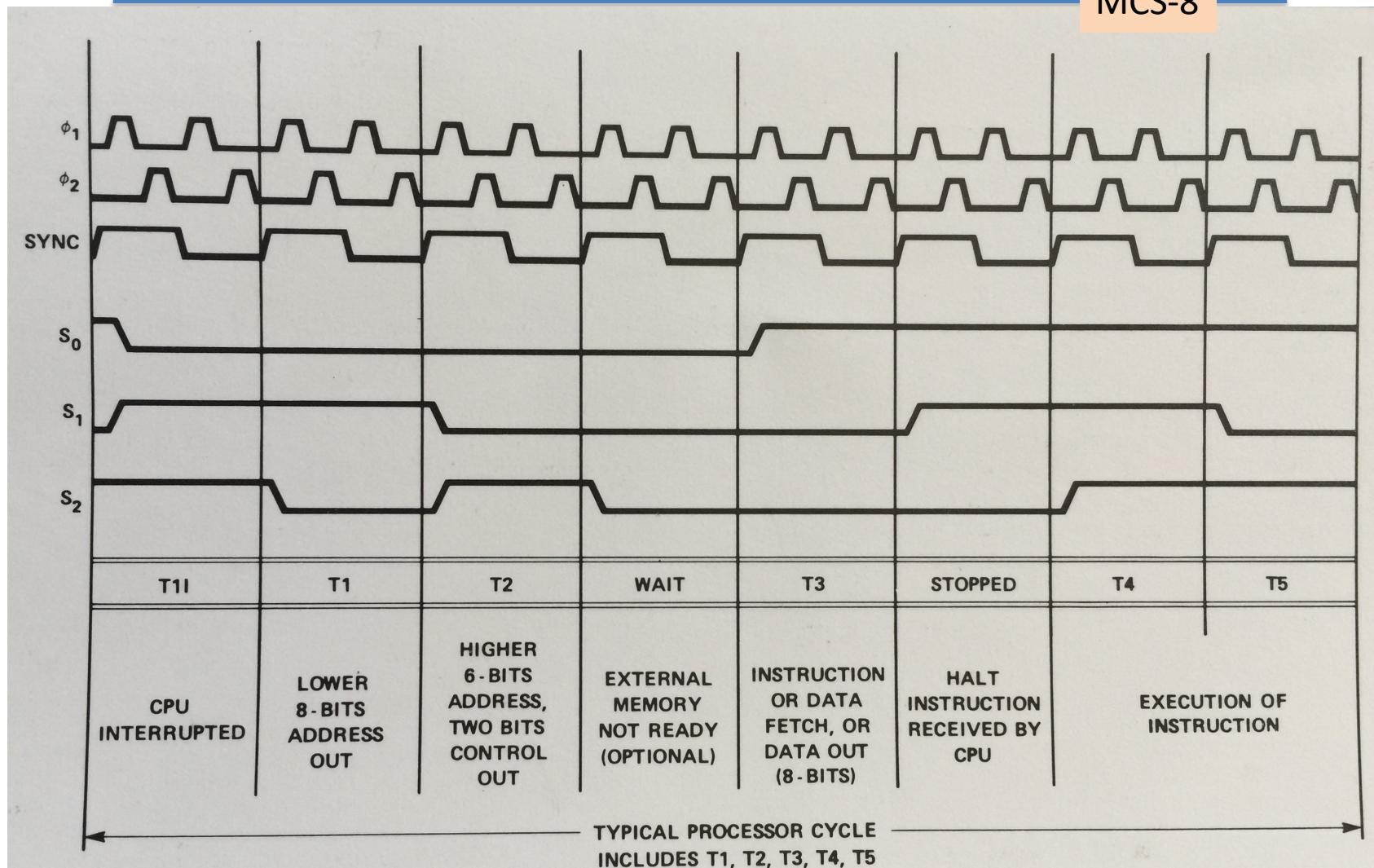
\*still exist



# CISC Instruction Cycle

## INTRO

MCS-8



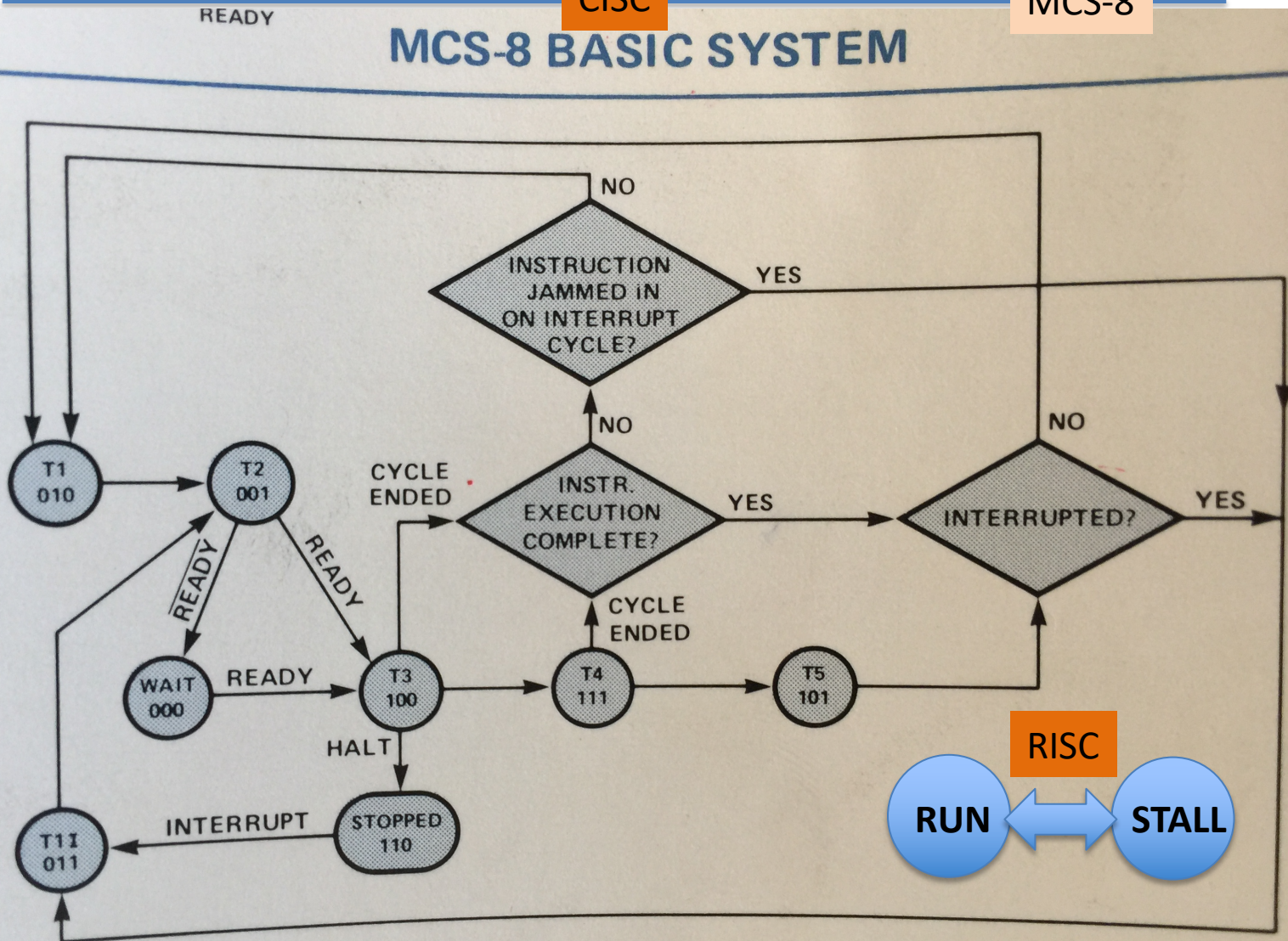
**MCS-8 BASIC INSTRUCTION CYCLE**



# CISC State Diagram

CISC

MCS-8



**CPU STATE TRANSITION DIAGRAM**

RISC

RUN

STALL

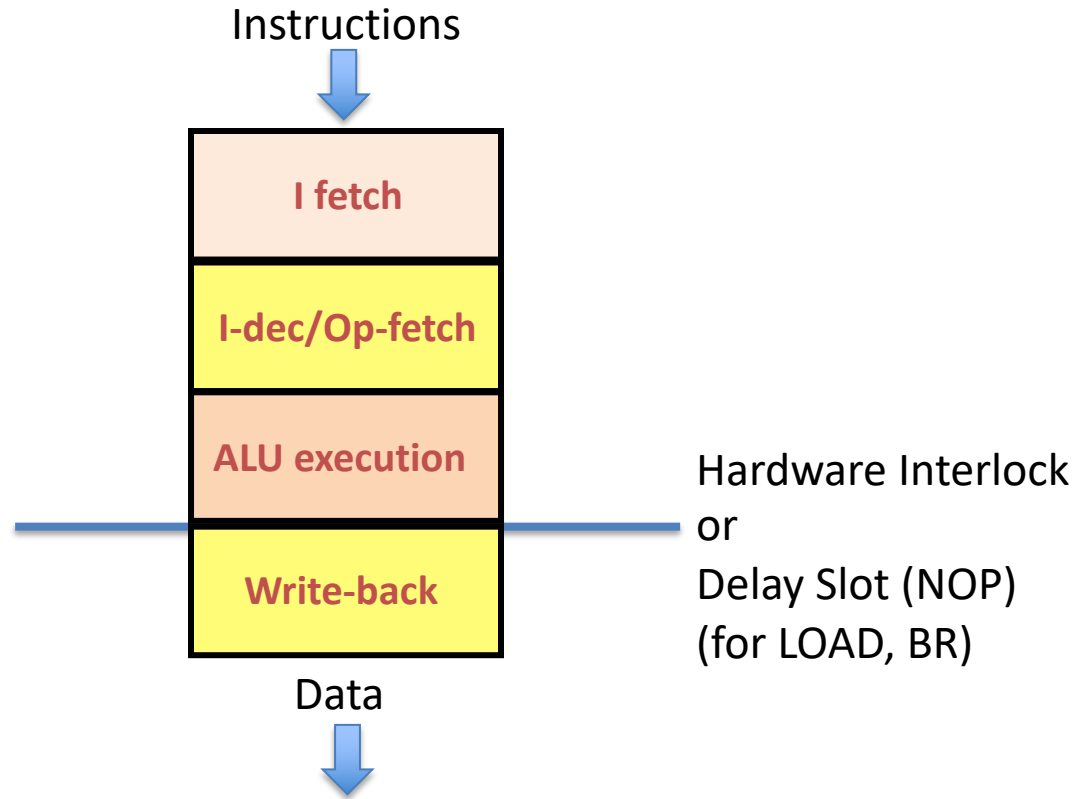
## *Reduced Instruction Set Architecture*

### ❖ Key Architecture of RISC

- Reduced ISA: small set of instructions
- Fast execution: single cycle only
- Reduced impact of memory
  - ✧ No microprogram (key change)
    - Instructions scale to vertical microinstructions (single-cycle)
    - eliminates ~30% chip area
  - ✧ LOAD-STORE (only) memory references
  - ✧ Full general register sets
  - ✧ Cache memory
    - On-chip
    - Multi-level
    - Harvard architecture – separate I and D
- Pipelining
  - ✧ 4 or 5 stages
  - ✧ Interlocks
    - Hardware (SPARC, 29K)
    - Software (MIPS): compiler manages pipeline scheduling



# RISC Pipelines



# CISC/RISC Pipelines

## Non pipelined

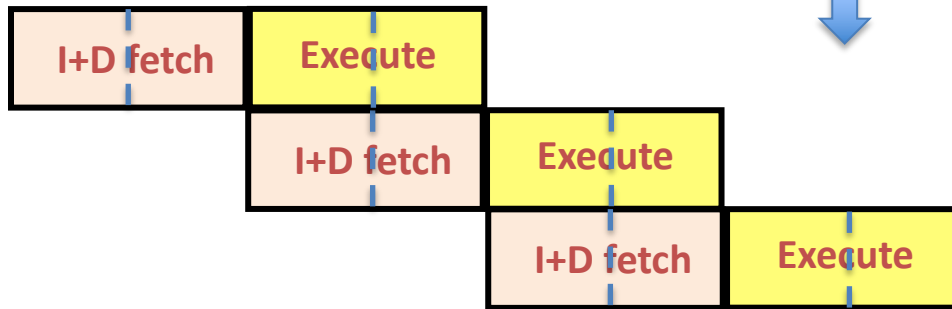
4-8 cycles per I  
i8008/M6800



One cycle

2-4 cycles per I i8088/M68000

## CISC Pipeline 2-stage



One cycle

Instructions



## RISC Pipeline 4-stage

R3000/SPARC/i960/29K/PPC

One cycle 1 cycle per I



Hardware Interlock  
or  
Delay Slot (NOP)  
(for LOAD, BR)

Data



# Embedded Control

## *Microprocessors*

For  
COMPUTING

- ❖ All 32/64-bit CPUs
- ❖ Large *data processing* applications
  - ◆ Employee records
  - ◆ Accounting
  - ◆ Payroll
- ❖ Operating systems (OS)
- ❖ “Apps” (applications)
  - ◆ PC/Mac
  - ◆ Mobile (phones, tablets)
  - ◆ Web apps
  - ◆ Cloud apps (SaaS)

Focus is **Memory**  
for large Data Files

*Large DRAM, Disk, Flash*

## *Microcontrollers*

For  
CONTROL

✧ *Real-time*  
✧ *All-in-one*

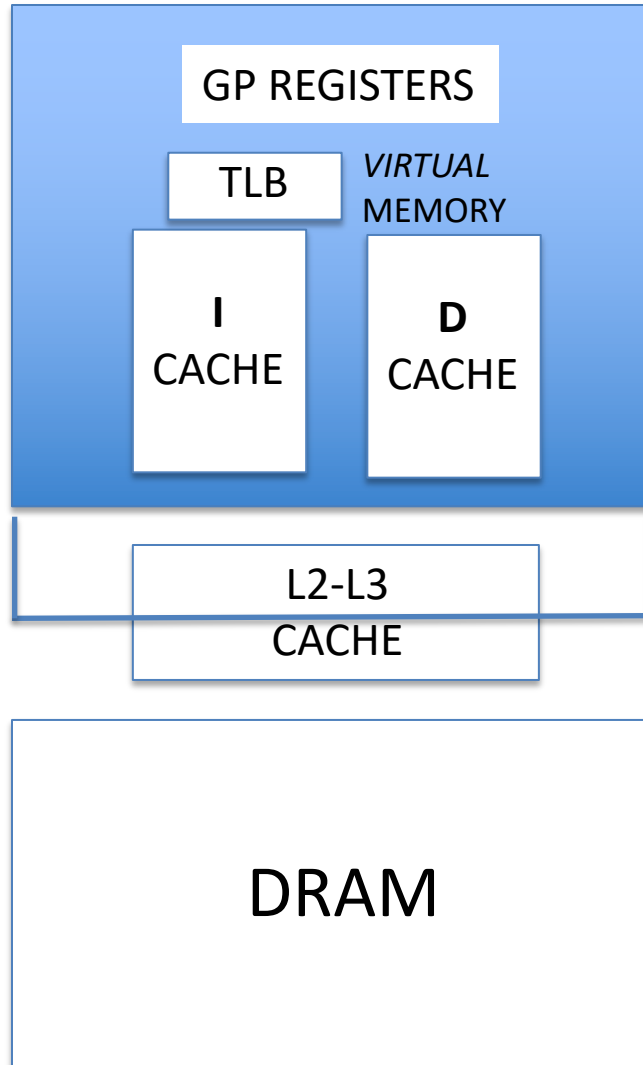
- ❖ Small *embedded control* applications (8-bit MCU)
  - ◆ Appliances
  - ◆ Disk controllers
  - ◆ Remote controllers
  - ◆ Garage/gate openers
- ❖ Medium *embedded control* (16-bit MCU)
  - ◆ User devices (iPods, phones, etc.)
  - ◆ Car/Airplane engine control
  - ◆ Car/Airplane braking & safety
  - ◆ Car transmission control
  - ◆ Home Automation (HAN)
- ❖ Large *embedded control* (32/64-bit MCU)
  - ◆ Car/Airplane entertainment
  - ◆ Car/Airplane navigation, systems management
  - ◆ Printers (MF)
  - ◆ Communications gear (WiFi, cable TV boxes)

✧ Tiny  
✧ Low power  
✧ Low cost

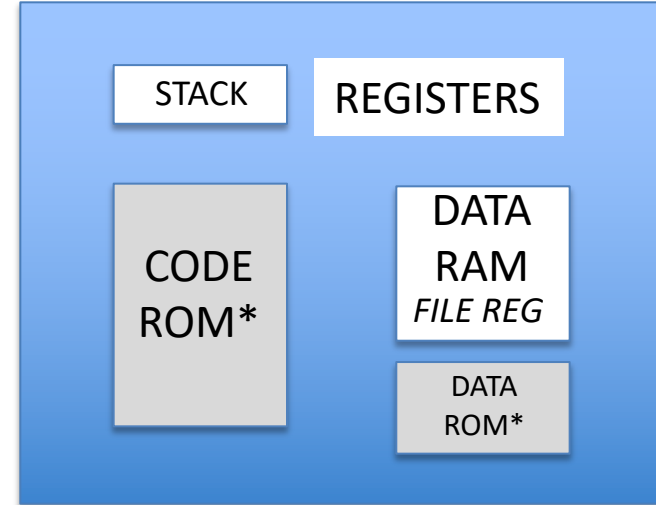
Focus is **I/O** – *Interrupts*

# Memory Models

## MICROPROCESSOR



## MICROCONTROLLER



***SMALL  
INTERNAL  
MEMORY***

*\*ROM contents  
must be  
"programmed"  
"burned", or masked*

***LARGE  
EXTERNAL  
MEMORY***

# Computer Architecture

---

## CPU Performance

# CPU Performance

$$\text{Time} = \text{Seconds/Program} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

$$\text{CPU time} = \text{Instruction count} \times \text{CPI} \times \text{Clock cycle time}$$

$$\text{Clock rate} = 1/\text{Clock cycle time}$$

$$\text{CPU time} = \frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}}$$



## INTRO

The following table summarizes how these components affect the factors in the CPU performance equation.

Hardware or software component	Affects what?	How?
Algorithm	Instruction count, possibly CPI	The algorithm determines the number of source program instructions executed and hence the number of processor instructions executed. The algorithm may also affect the CPI, by favoring slower or faster instructions. For example, if the algorithm uses more divides, it will tend to have a higher CPI.
Programming language	Instruction count, CPI	The programming language certainly affects the instruction count, since statements in the language are translated to processor instructions, which determine instruction count. The language may also affect the CPI because of its features; for example, a language with heavy support for data abstraction (e.g., Java) will require indirect calls, which will use higher CPI instructions.
Compiler	Instruction count, CPI	The efficiency of the compiler affects both the instruction count and average cycles per instruction, since the compiler determines the translation of the source language instructions into computer instructions. The compiler's role can be very complex and affect the CPI in varied ways.
Instruction set architecture	Instruction count, clock rate, CPI	The instruction set architecture affects all three aspects of CPU performance, since it affects the instructions needed for a function, the cost in cycles of each instruction, and the overall clock rate of the processor.

# Computer Architecture

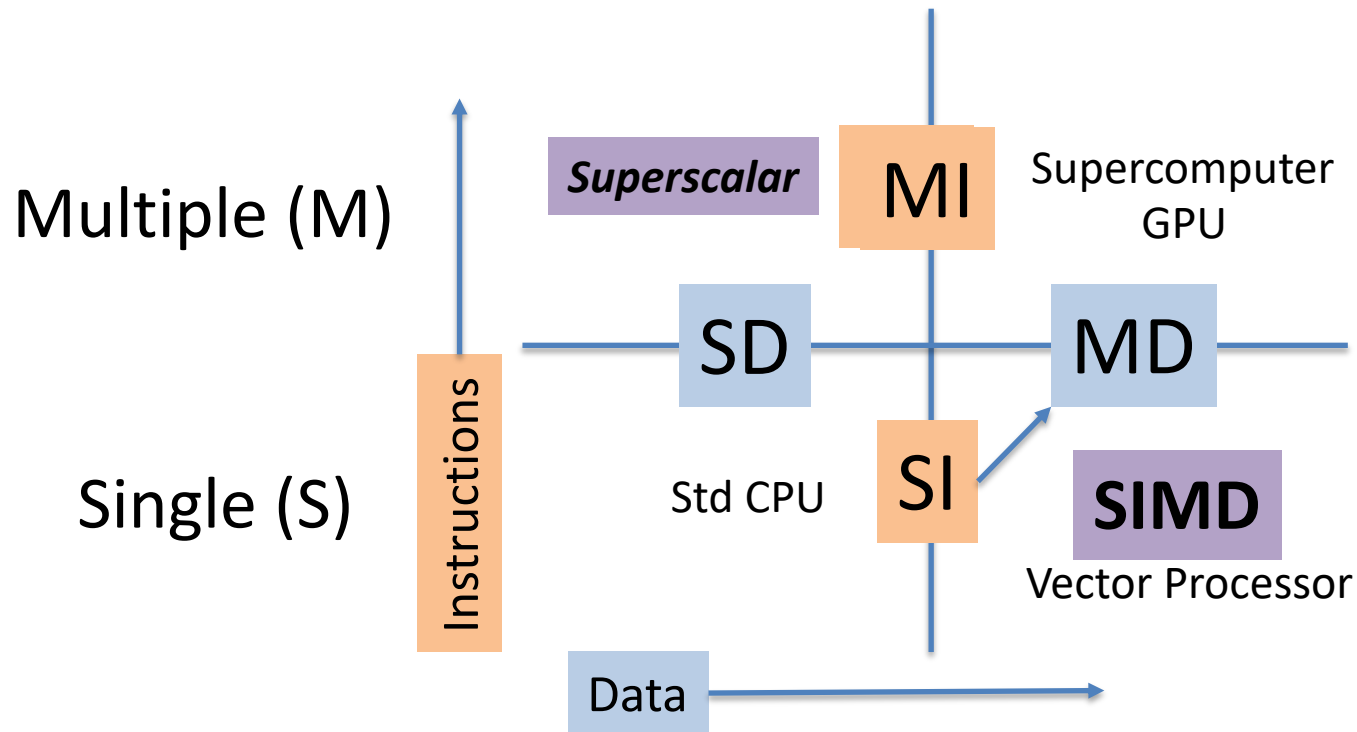
---

## Parallelism

# I-D Parallelism: SIMD

## Flynn Partition

Michael J. Flynn paper (U Illinois (UIUC), Ca 1969)



# Instruction Level Parallelism

---

## ❖ Super- *Pipelining*

- ☐ SISD (single instruction)
- ☐ Split some pipeline stages
- ☐ Faster clock cycle → higher *throughput (mips)*
- ☐ Affect CPI?

## ❖ Super- *Scalar*

- ☐ MISD (multi-instruction)
- ☐ Multiple pipelines → each with own ALU
- ☐ Requires compiler to schedule instruction streams

## ❖ Multi- *Threading*

- ☐ SISD (single instruction)
- ☐ Multiple *control threads*
- ☐ Requires programmers to schedule *control threads*

## ❖ Multi- *Core*

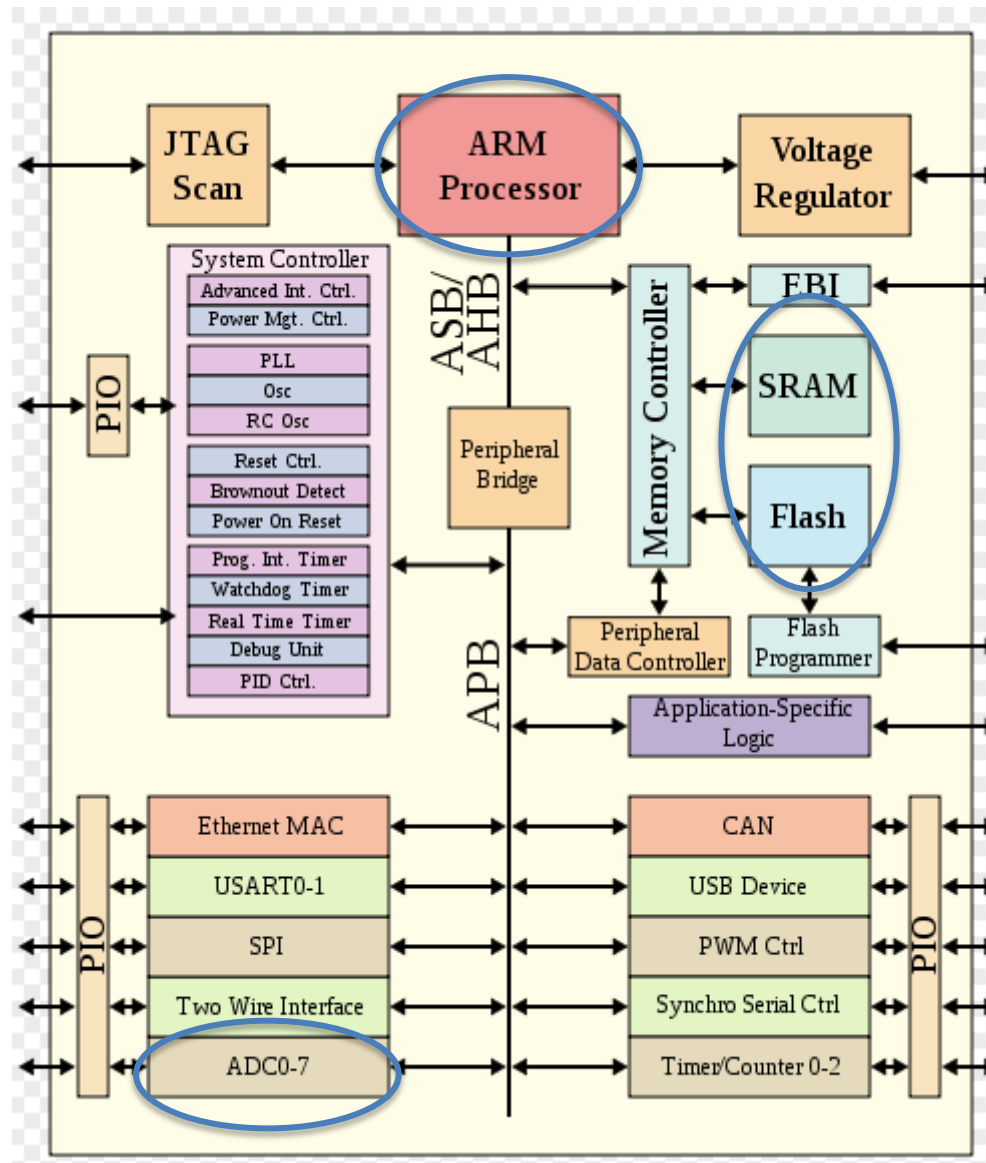
- ☐ Classic Parallelism: multiple copies of the whole CPU
- ☐ **Multiple L1 caches** (one per core)

# Computer Architecture

---

## Other Hardware (Peripherals)

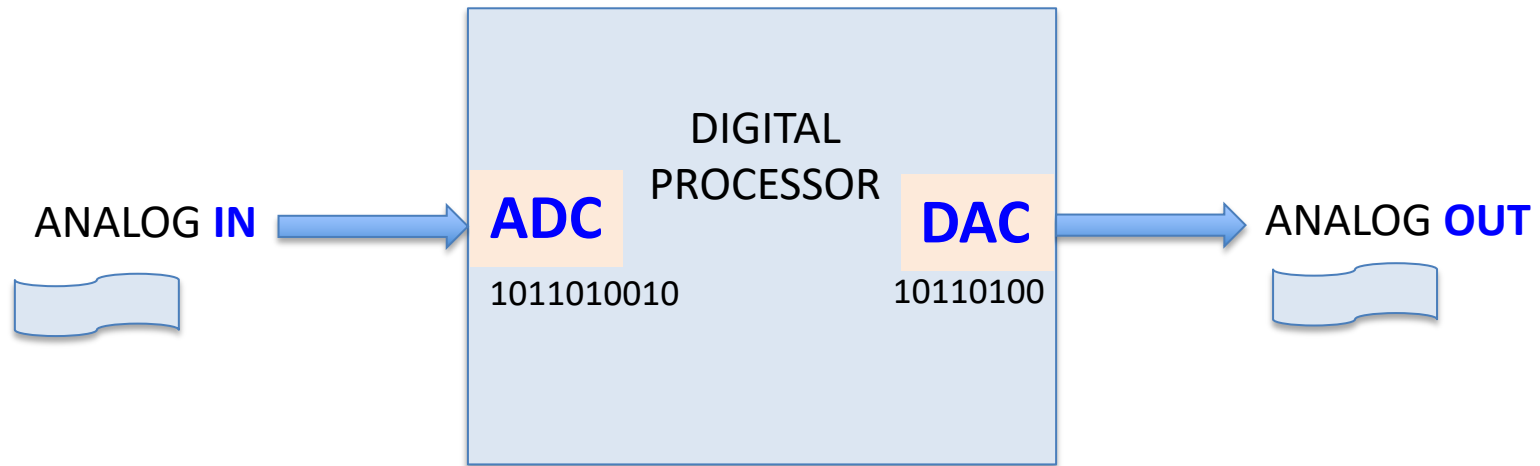
# Hardware-ARM SoC





# Data Conversion

*Embedded Control lives in an ANALOG world*



## ❖ ADC

- ✧ Typ 8-14 bits (resolution)
- ✧ Flash or SAR

## ❖ DAC

- ✧ Byte (8-bit)
- ✧ Resistor ladder

# Software

Computer  
Science

## Software Models: Levels & Layers

# Realms of Software

~70% of all software

## ❖ Applications

- ☐ Desktop
- ☐ Mobile (Apps)
- ☐ Web

## ❖ Web

- ☐ Markup
- ☐ Applications
- ☐ SQL databases

## ❖ Embedded Control

- ☐ Small (8-bit)
- ☐ Medium (16-bit)
- ☐ Large (32/64-bit)

❖ APIs (Frameworks)

❖ Client-Server model

❖ Language “stacks” (e.g., LAMP)

❖ From TV remotes to

❖ Autonomous cars and

❖ Robots

### ➤ Common required properties

- Performance
- Reliability (bug free)
- *Security*

# Software *Levels*

High-Level

```
Imports System.Drawing.Printing
Public Class Form1
    Inherits System.Windows.Forms.Form
    '**system constants
    Public Version As String = "Version x.x"
    Dim DataVer As String 'ver # in file
    MyBase.Load
        copyrt.Text = "Copyright(c) 2007-12"
    DemoLab.Visible = DEMO
    boxcolorY = CatBox.BackColor
```

*Human* readable  
(.htm, .js, .php, .vb files)

Assembly

```
LD R1,X
ADD R1,R2,R3
```

hybrid  
(.asm files)

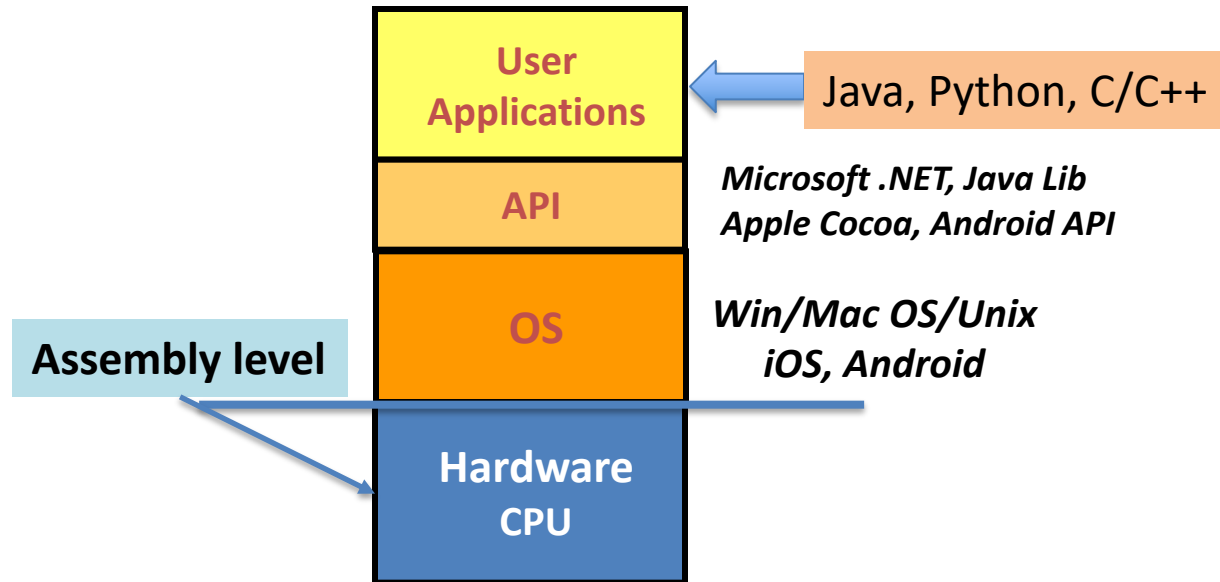
Machine  
(Binary)

```
1011010010101101
```

*Machine* readable  
(.exe files)

# Software *Layers*

Simple View



# Com Protocol Layers

## 7-Level OSI MODEL

### of Protocol Layers

### IN COMMUNICATIONS SYSTEMS

LEVEL

EXAMPLES

DIGITAL

ANALOG

7

Application

Office applications suite, Adobe Acrobat  
Internet applications: HTTP, FTP, POP, SMTP

6

Presentation

SSL, encryption, compression

5

Session

connections

4

Transport

TCP, UDP, TLS

3

Network

IP addressing &  
routing

2

Data Link

MAC (Media Access Control)

1

PMI

PHYSICAL

PMD

PHY-  
CDR  
Transceiver  
Optical-  
Laser diode/LED, Photodetector, TIA, PA



# Hardware-Software *Layers*

## 7-Level

STACK HIERARCHICAL MODEL OF  
**LEVELS OF DESIGN**  
OF DIGITAL SYSTEMS

LEVEL

EXAMPLES

**SOFTWARE**

Middleware

Firmware

PROCESSOR  
ARCHITECTURE

**HARDWARE**

DIGITAL

ANALOG

7

6

5

4

3

2

1

Data

Applications

API

OS

Microprogram

Processor  
ISA

LOGIC

--- PHYSICAL ---

.doc, .xls, .sql, .csv, .txt files

Microsoft WORD, Excel & "apps"

Microsoft .NET, Java Lib  
Apple Cocoa, Android API

Win/Mac OS/Unix  
iOS, Android

Intel/AMD Pentiums  
MIPS, Sparc

Logic Design

Communications

(NOTE: *FIRMWARE* is any embedded software, such as microprograms, monitors, real-time executives, etc.)

# Problem Solving

---



Computer  
Science

## Top-Down Analysis & Design

# Problem Solving

Problem → Requirements → Design → Implementation



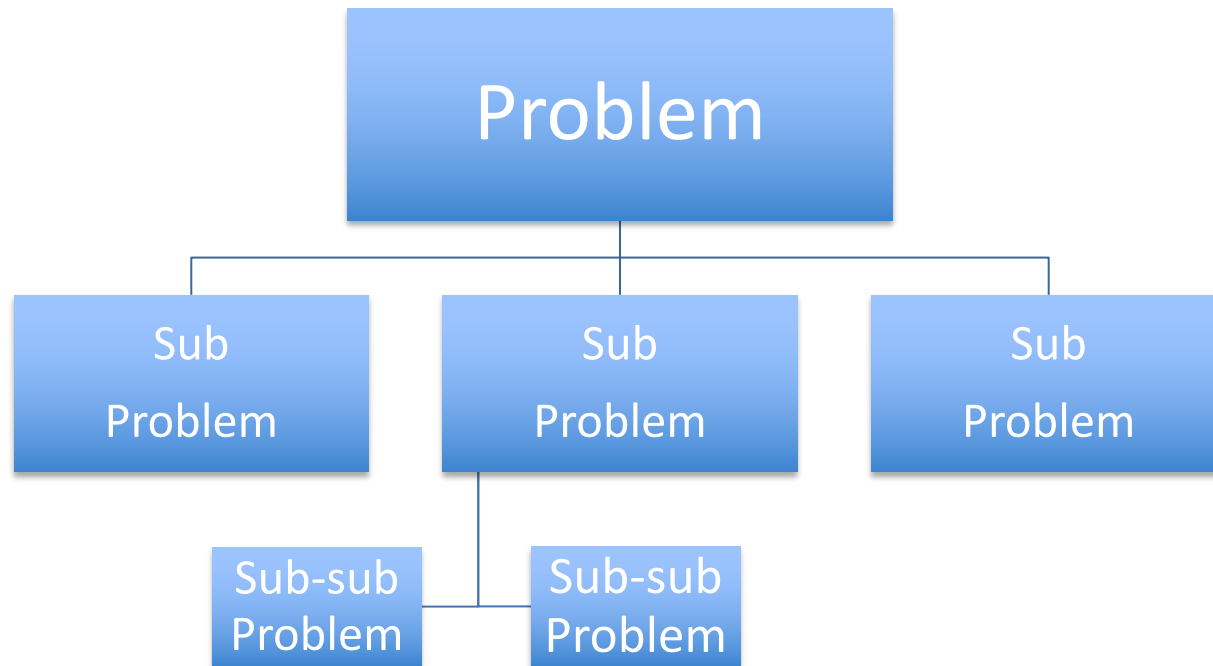
SDLC

## ❖ Requirements

1. Function 1
2. Function 2
3. Function 3
4. Function 4
5. Function 5
6. Feature 1
7. Feature 2

# Top-Down Analysis

“Problem” → “Requirements”



Each Sub Problem → Requirements/Function

# Top-Down System Design

“System” = Hardware + Software

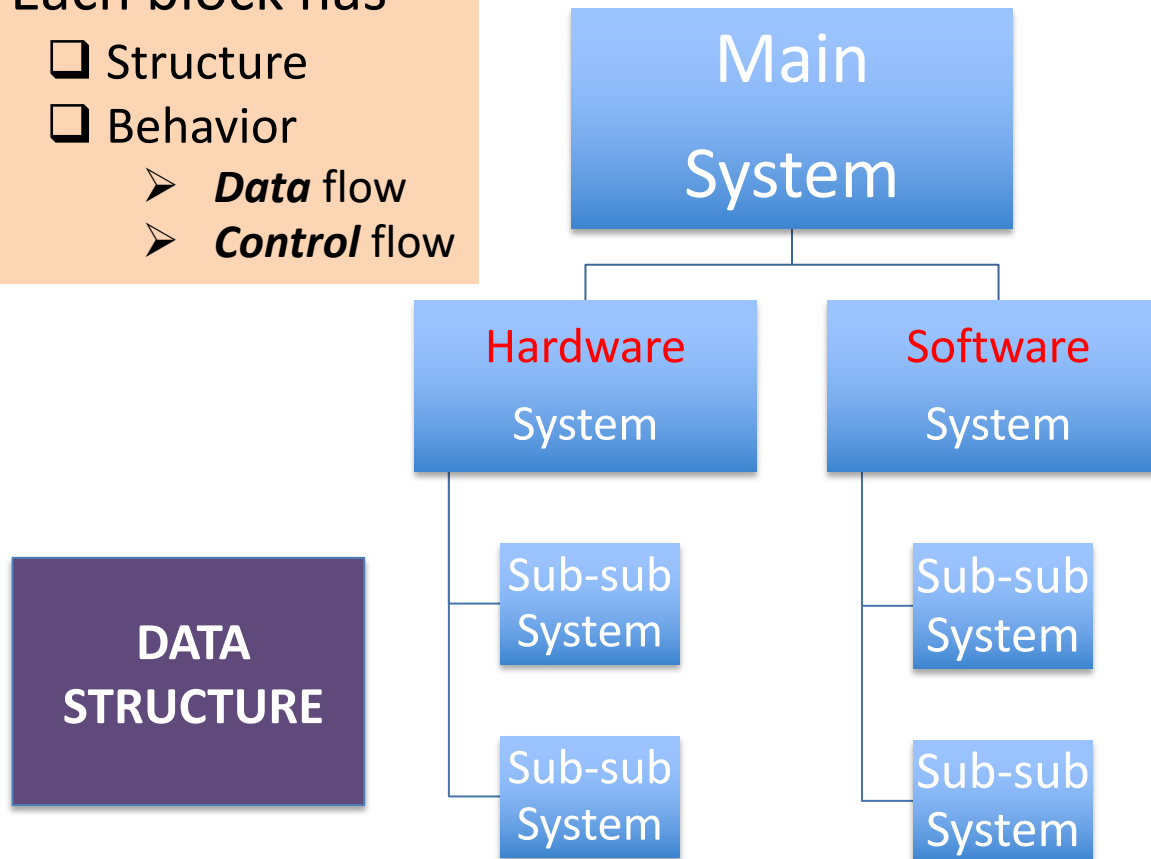
❖ Each block has

☐ Structure

☐ Behavior

➤ **Data** flow

➤ **Control** flow



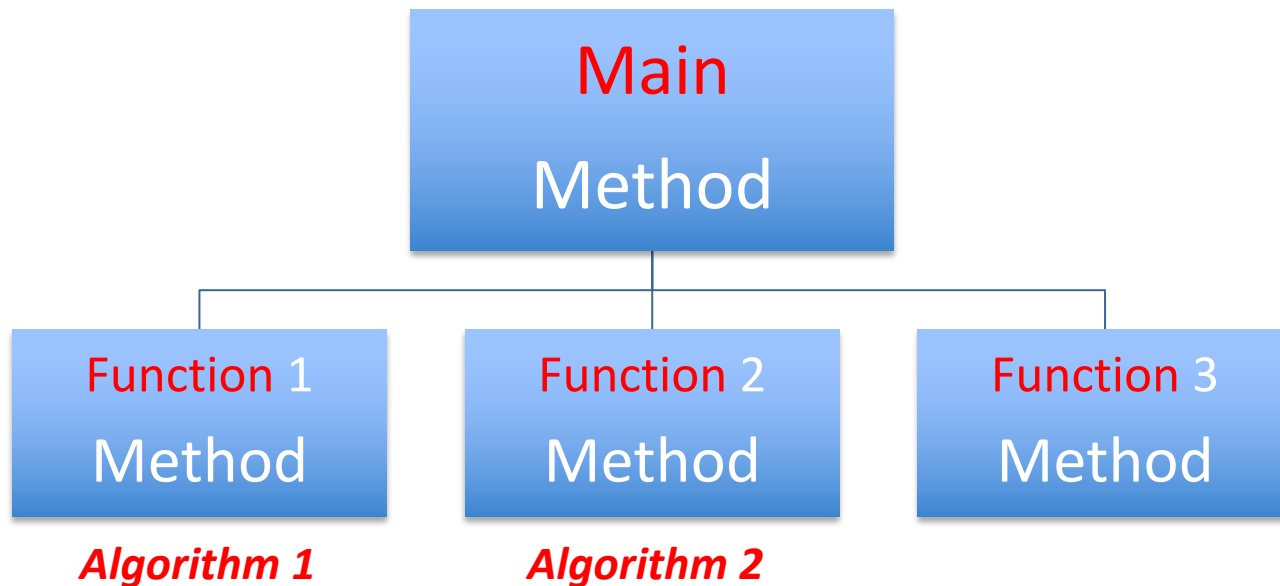
**Structure** Mapped to Requirements



# Software Structure

Tree structure

***Structure*** Mapped to ***Requirements***



➤ Not all ***Functions*** are implemented by ***Algorithms***

# Software Structure

---

Computer  
Science

Programs

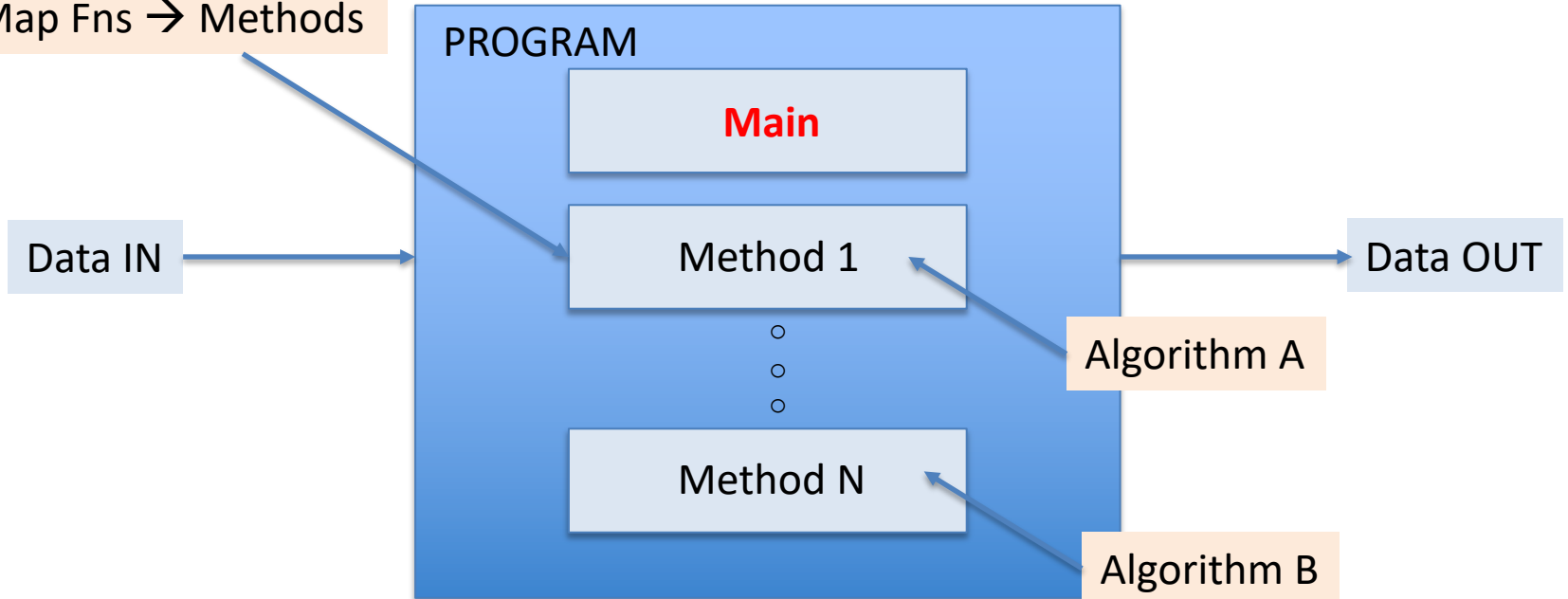
# Programs & Algorithms

Nested Structure

Requirements

Procedure → Process

➤ Map Fns → Methods



➤ 1 Algorithm per *function* or *method*

# Program Debugging

## Errors and Bugs

Requirements

Data IN

PROGRAM

you find & fix

BUGS

ERRORS

Data OUT

- ❖ compiler finds
- ❖ you fix

- ❖ You find

### ❖ *ERRORS*

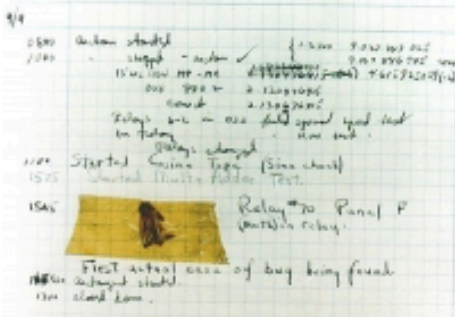
- ❑ syntax → compiler
- ❑ semantic/logic → debug
- ❑ exceptions → handlers

### ❖ *BUGS*

- ❑ *DEBUG*
- ❑ *in situ* → IF-THEN
- ❑ exception handlers

# 1<sup>st</sup> Bug

## Technology First Computer Bug



Handwritten log book entry from 1947, dated 9/9. The entry describes the first actual case of a bug being found. It includes a photograph of a large moth, approximately four inches in wing span, which was found in one of the relays and beaten to death. The text is written in cursive and includes the following details:

- 9/9
- 0800 Machine started
- 1000 - stopped - action ✓
- 1500 low MP - MP
- 000 000 - 2 1200000
- 0000 0000
- Relays are in 000 full speed and not in today
- Relays changed
- 1100 Started Machine Tape (Bina chart)
- 1500 Started Machine Tape Test
- 1500 Relay to Panel P
- 1500 Machine relay
- First actual case of bug being found
- 1500 Machine started
- 1500 closed down

And one night she (Mark II) conked out and we went to look for the bug and found an actual large moth, about four inches in wing span, in one of the relays beaten to death, and we took it out and put it in the log book and pasted Scotch tape over it.

EDN (1947)

(Click image to view full size)

❖ 1947

"It was over in another building, and the windows had no screens on them and we were working on it at night, of course, and all the bugs in the world came in. And one night she (Mark II) conked out and we went to look for the bug and found an actual large moth, about four inches in wing span, in one of the relays beaten to death, and we took it out and put it in the log book and pasted Scotch tape over it."

# Software Engineering

---

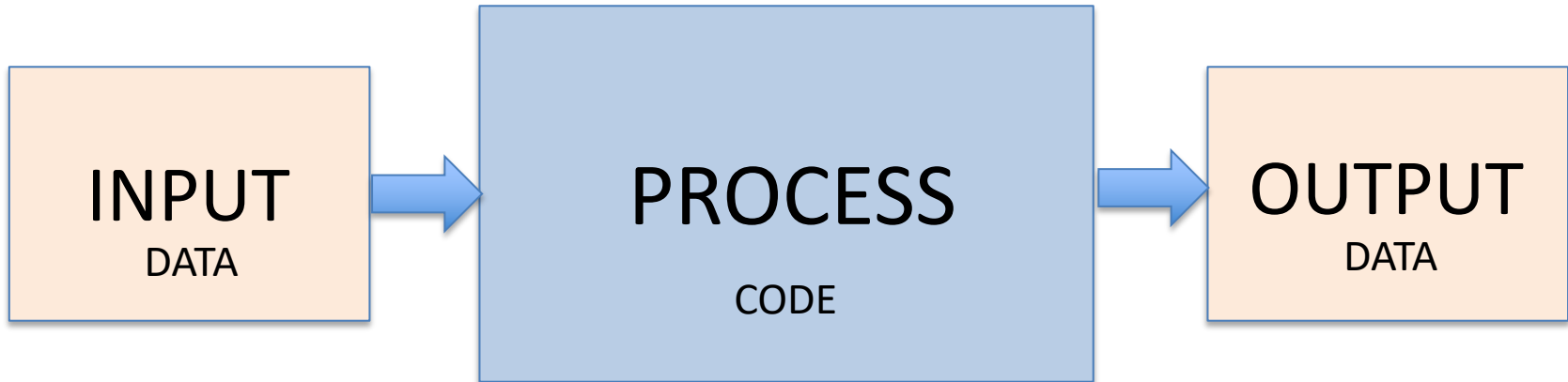
Computer  
Science

## Models

We will use these



# IPO Model



# Code + Data

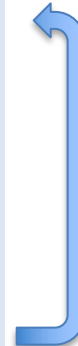
## Software

### ❖ Code structure

- ❑ Macro
  - Classes
  - Methods
- ❑ Control (Micro)
  - *Conditional* blocks
    - If
    - Switch - Case
  - Loops
    - For
    - While

### ❖ Data structure

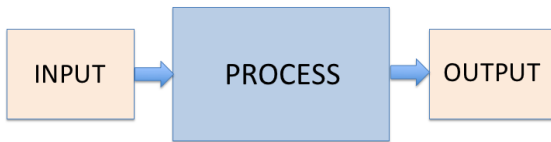
- ❑ In Code
  - **Arrays**
  - Array-Lists
  - Enums
  - Collections
- ❑ In Files
  - **CSV files**
  - Databases



### ❖ Data types

- ❑ Numeric
  - Integer
  - Floating-point
- ❑ Non-Numeric
  - Logic (Boolean)
  - Characters
  - Strings

# Sandwich Model



## ❖ Bread (encapsulation)

- ❑ Classes
- ❑ Methods

➤ STRUCTURE

## ❖ Greens (passing data)

- ❑ I/O
- ❑ Parameter passing

➤ Input/Output

## ❖ Cheese (blocks/constructs)

- ❑ Conditional structures
  - IF-THEN-ELSE
  - SWITCH-CASE
- ❑ Loops

➤ CONTROL

## ❖ MEAT (statements → gist)

- ❑ Arrays
- ❑ Strings
- ❑ Expressions
  - Arithmetic
  - Logical

➤ DATA

# Software Engineering

---

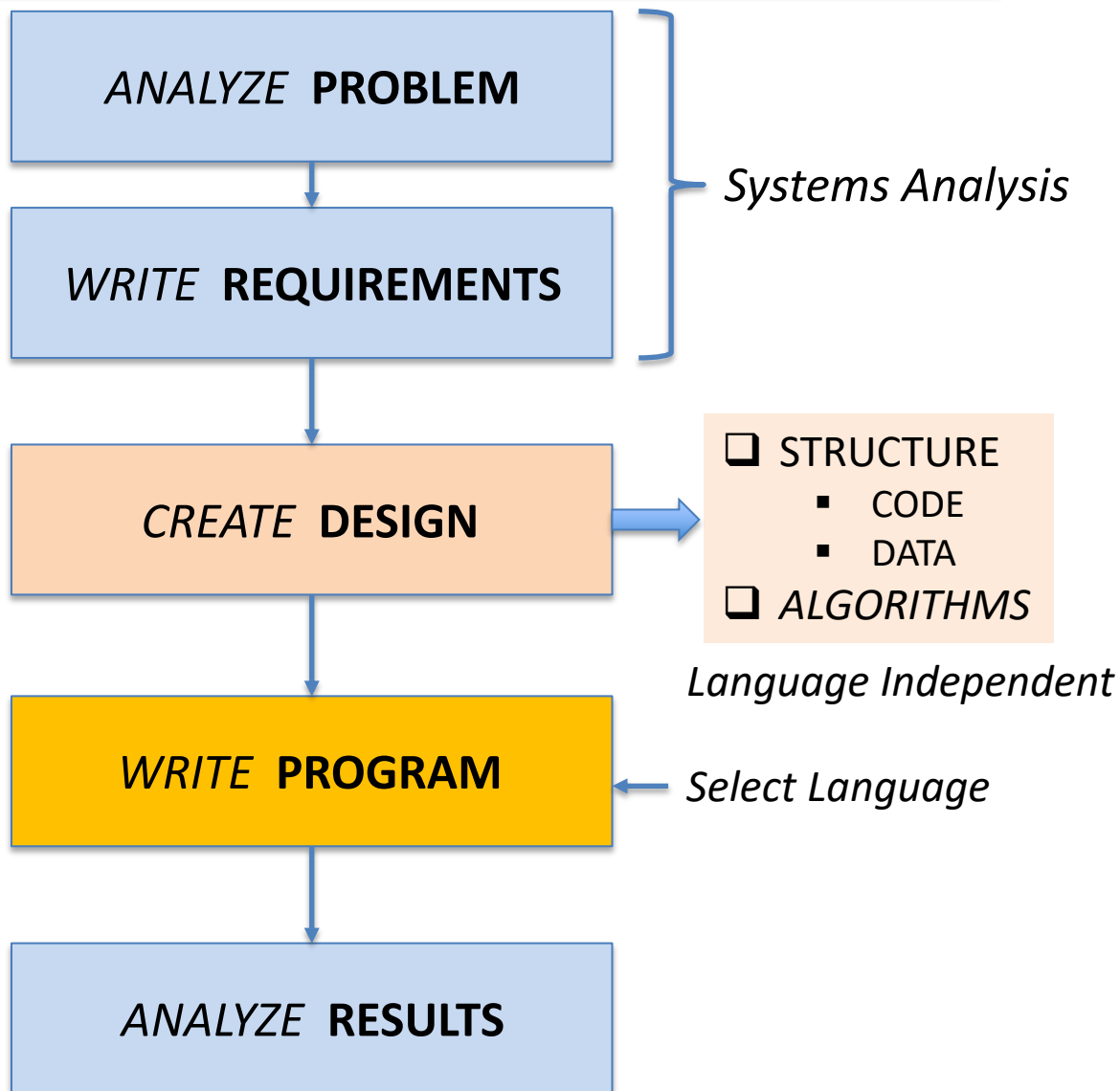


Computer  
Science

SDLC

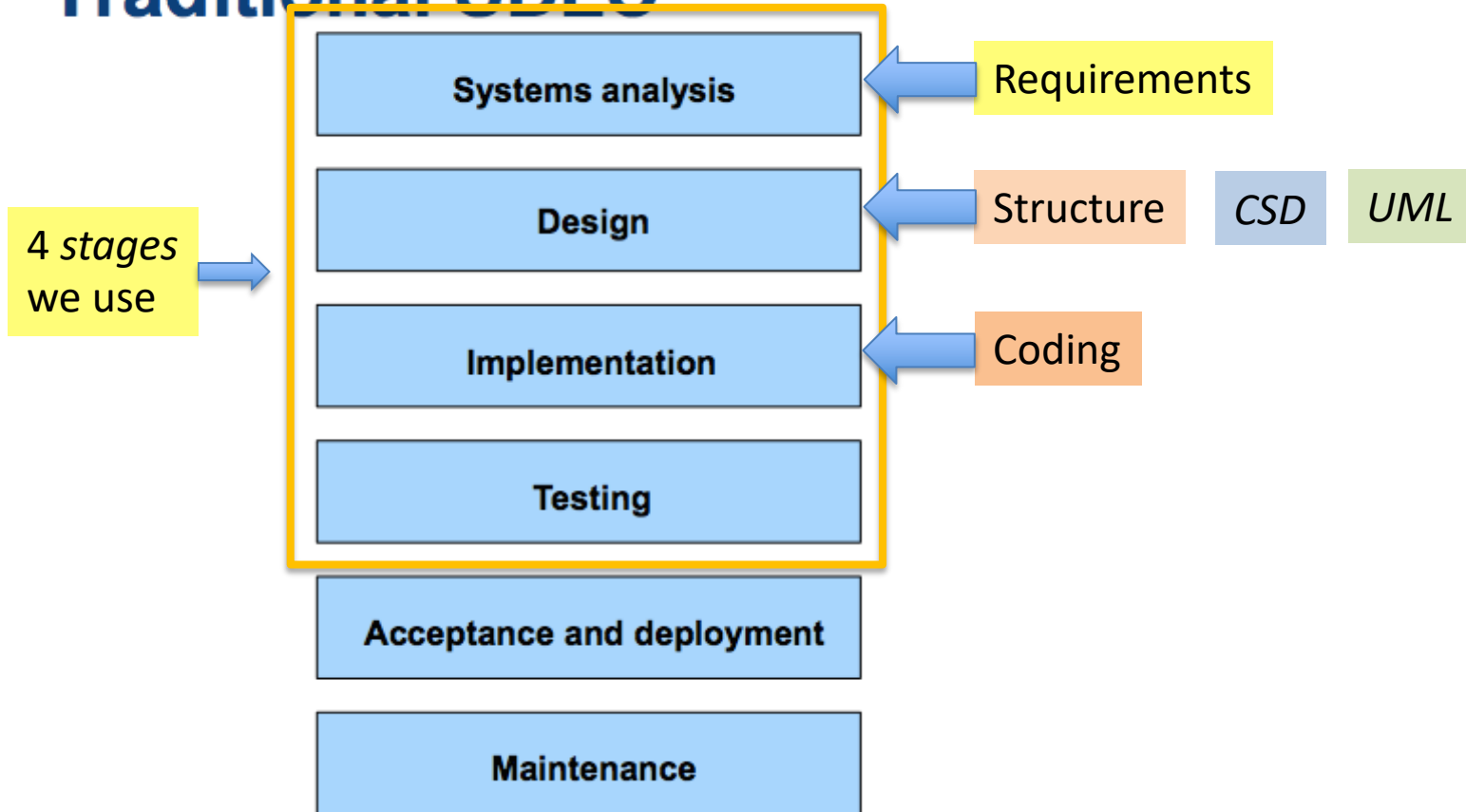
# Software Development

Development Procedure  
(similar to **SDLC**)



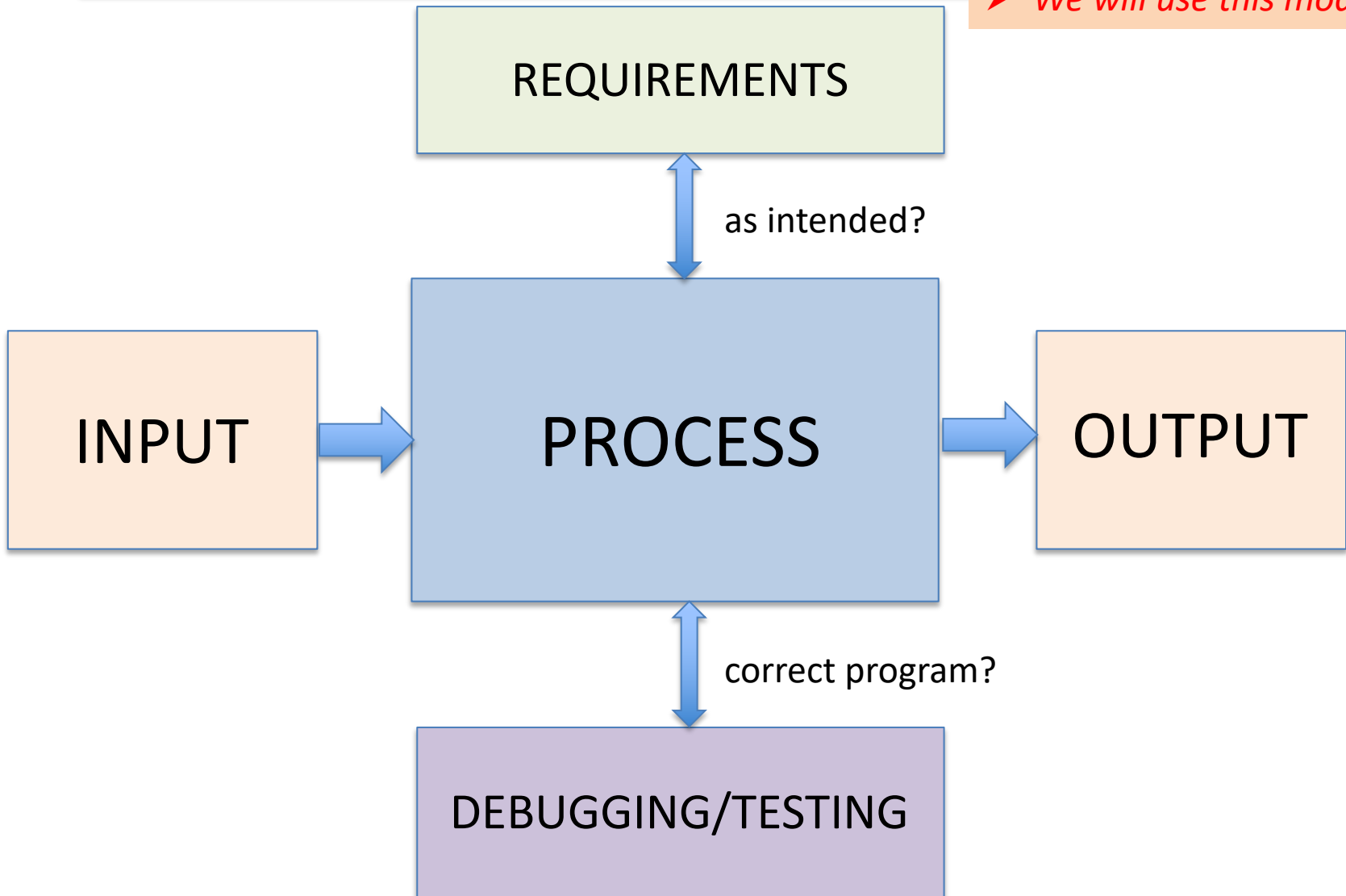
## SOFTWARE DEVELOPMENT LIFE CYCLE

### Traditional SDLC



# SDLC + IPO

➤ *We will use this model*

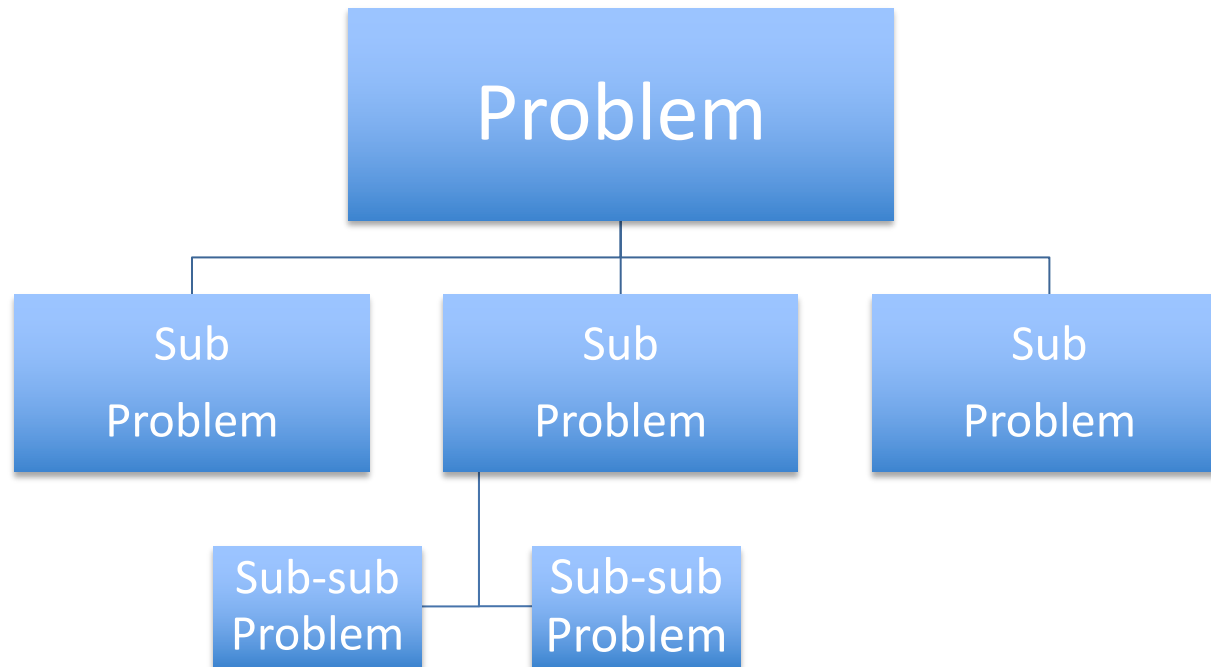




# SDLC – Requirements

SDLC – *System Analysis*

Problem (Work statement) → **Requirements**



# SDLC – Requirements

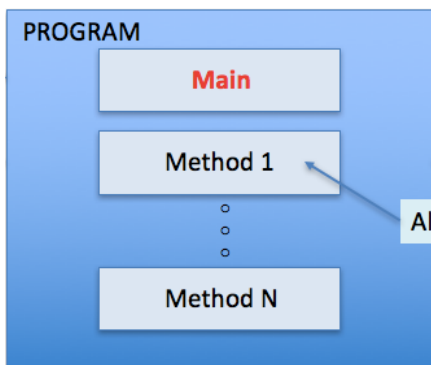
## SDLC – System Analysis

### ❖ Types of Requirements

- ☐ ***Functions*** (“what”)
- ☐ ***Features*** (“how”)
- ☐ Performance (speed)
- ☐ Cost
- ☐ Time/schedule

# SDLC – Code Design

## SDLC – *Design*



```
void main( ) {
//call subs
}
```

Main  
Program

Sub  
Program

Sub  
Program

Sub  
Program

```
public int sub1( arg_list) {
int fn1
fn1 = formula1
return(fn1)
}
```

Sub-sub  
Program

```
#include <stdio.h>
#include <iostream>
#include <stdnamespace>
```

C/C++

### ➤ Include/Import LIBRARIES

```
import javax.swing.JOptionPane
import javax.*
```

Java

# SDLC – Data Design

SDLC – *Design*

**DATA  
STRUCTURE**

DATA FILE A

DATA FILE B

DATA FILE C

DATA FILE D

-OR-

**DATABASE**

# SDLC – Testing

## ❖ Reliability

### ❑ Testing

#### ➤ Unit testing

#### ➤ Black-box

### ❑ Error handling

#### ➤ Compile time

##### ✧ Warnings

##### ✧ Errors

#### ➤ Run time

##### ✧ TRY-CATCH blocks (Avoid crashes)

### ❑ Debugging

#### ➤ Debug Tools (breakpoints, watch variables)

#### ➤ Embedded debug code

SDLC – *Testing*

```
stmt1  
stmt2  
stmt3  
stmt4  
stmt5  
stmt6  
stmt7
```

Compile & run  
Compile & run  
Compile & run

## Technology First Computer Bug



## ❖ Security

### ❑ Top 10 vulnerabilities

### ❑ Monitor & Detect

### ❑ Plug security holes

# Software Engineering

---



Computer  
Science

## Debugging & Testing

# Debugging

---

## □ Debugging

### ➤ Debug Tools

- Breakpoints
- Watch variables
- Activation stack

### ➤ Embedded debug code

- Breakpoints (implicit)
- Print/log variables
- Print/log activities
- Use “\$DEBUG” switch



# Code-Fix Process

Tsui & Bernal (2014), *Essentials of Software Engineering*

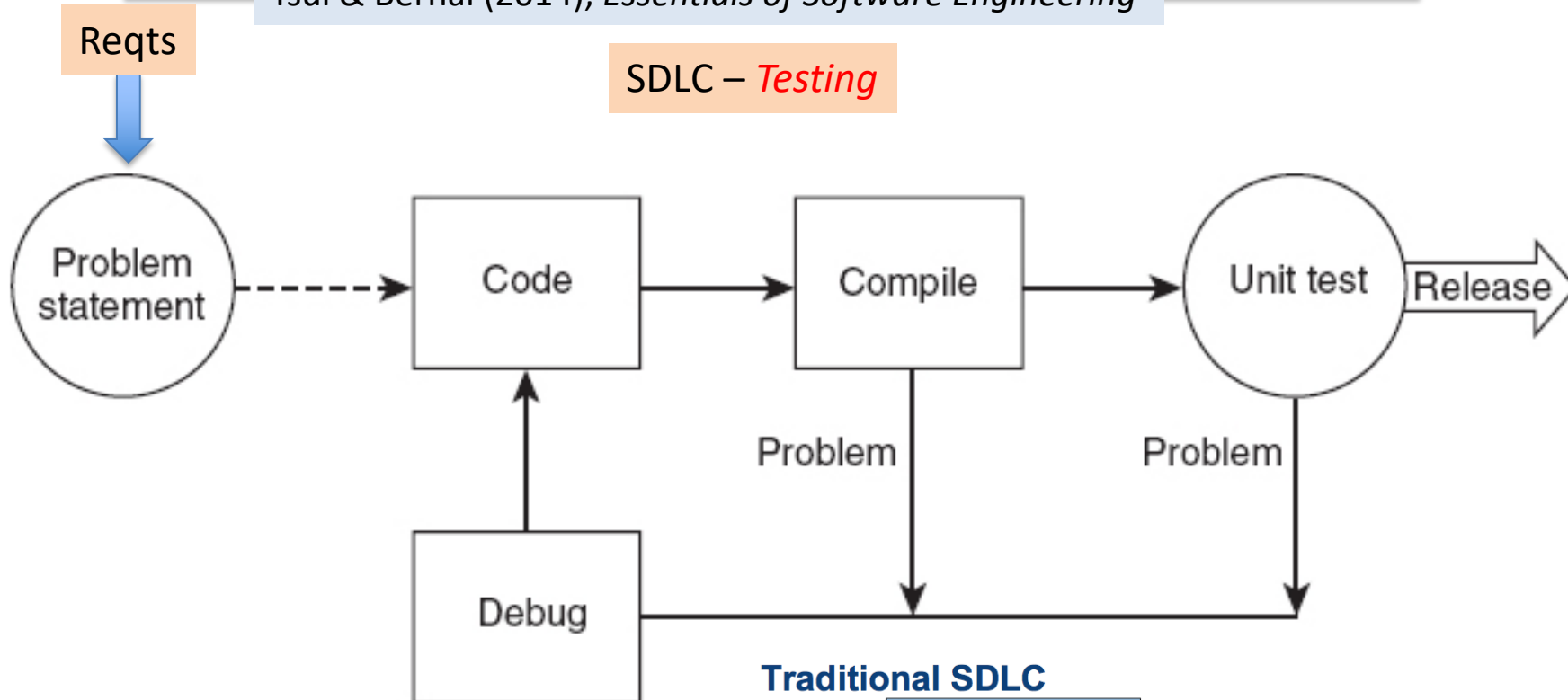
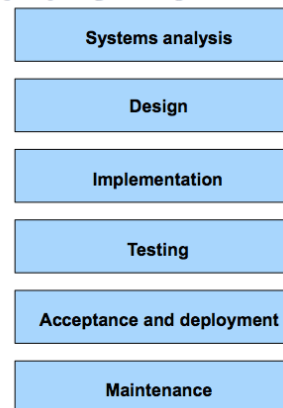


Figure 4.1: A simple process.

## Traditional SDLC



# Incremental Process

## INTRO

Tsui & Bernal (2014), *Essentials of Software Engineering*

SDLC – *Testing*

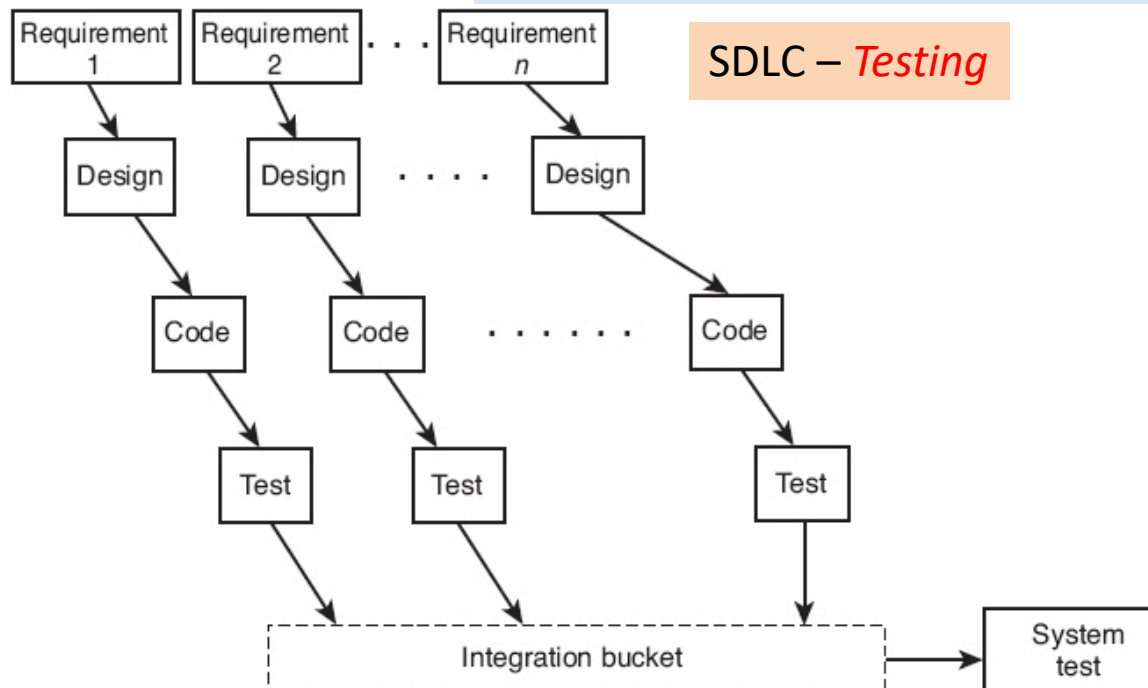


Figure 4.3: A multiple-components incremental model.

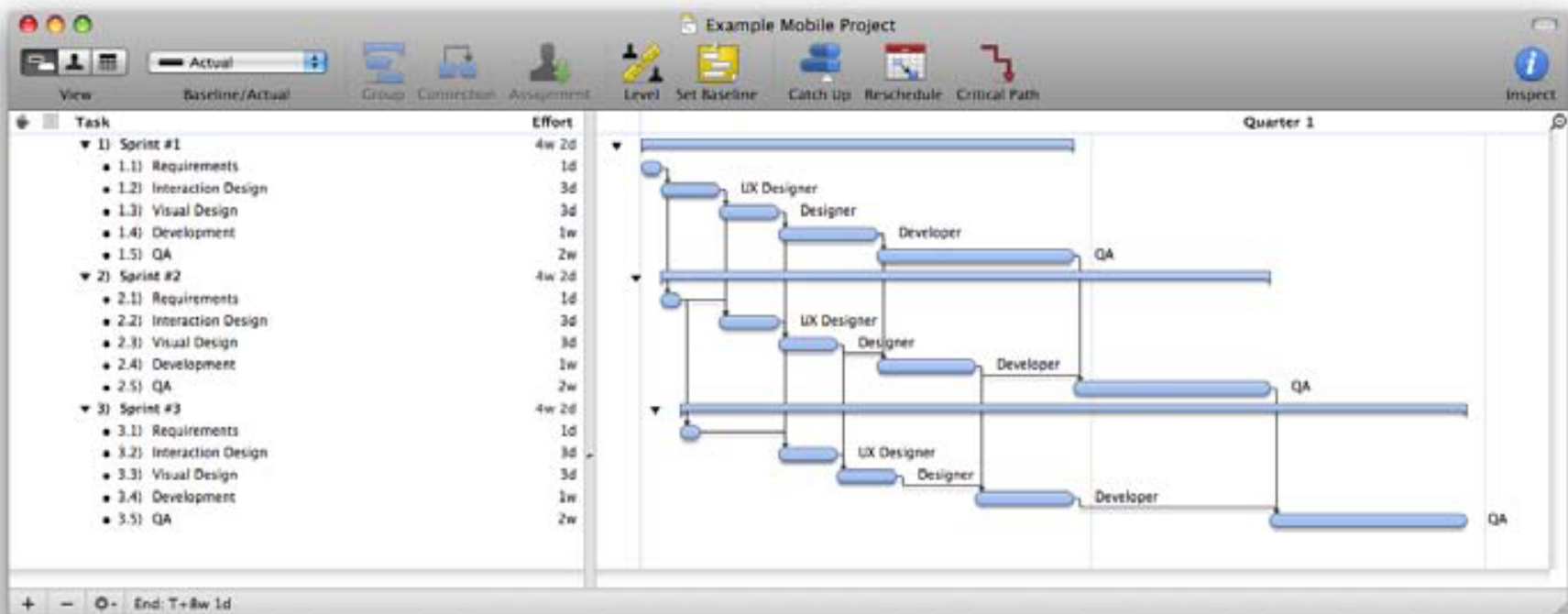


Figure 4.4: A multiple-release incremental model.

# Estimating Testing Effort

SDLC – *Testing*

❖ Device testing takes 2 to 4 times development effort



# Test Plan

## SDLC – *Testing*

### ❖ Functional tests

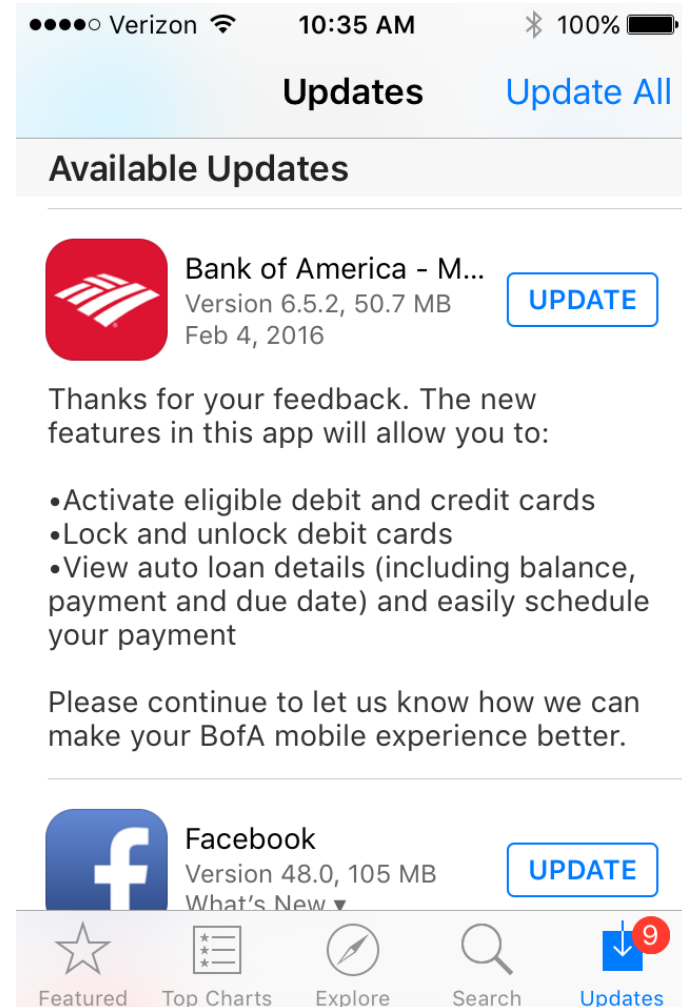
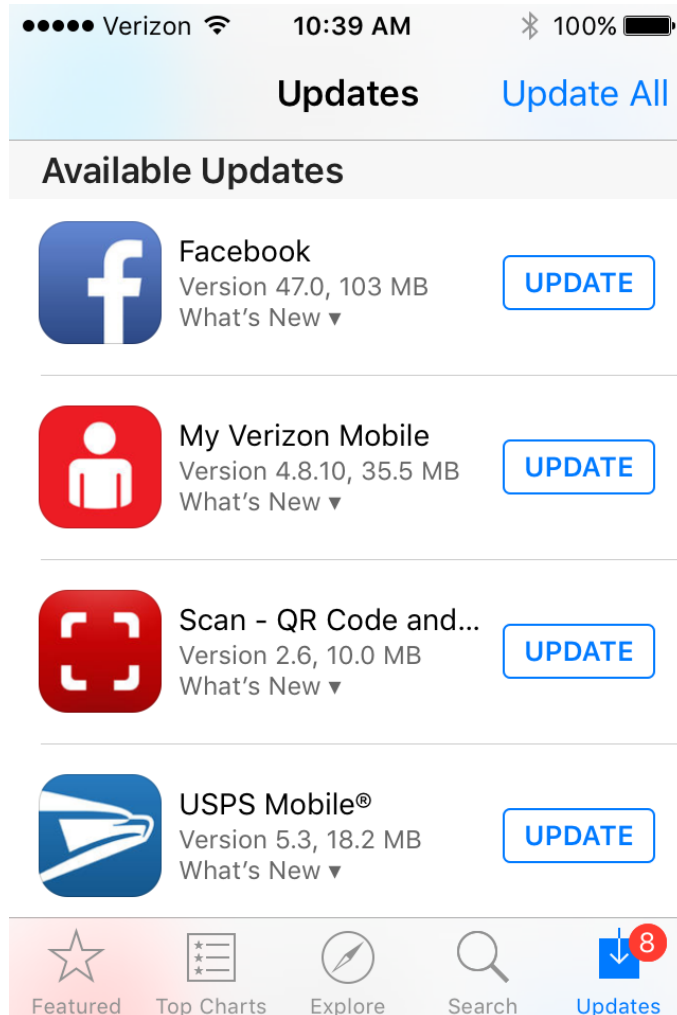
- Based on feature list

### ❖ Context tests (especially mobile)

- How does the **user experience** render on the device?
- Does it **load** quickly and correctly?
- Can you use the **physical features** of the device as intended?
- Does it **terminate** correctly?
- What happens when the device loses **connection**?
- Does the application work when hopping from **cell** tower to cell tower?

# SDLC – Revs/Updates

## SDLC – *Maintenance*



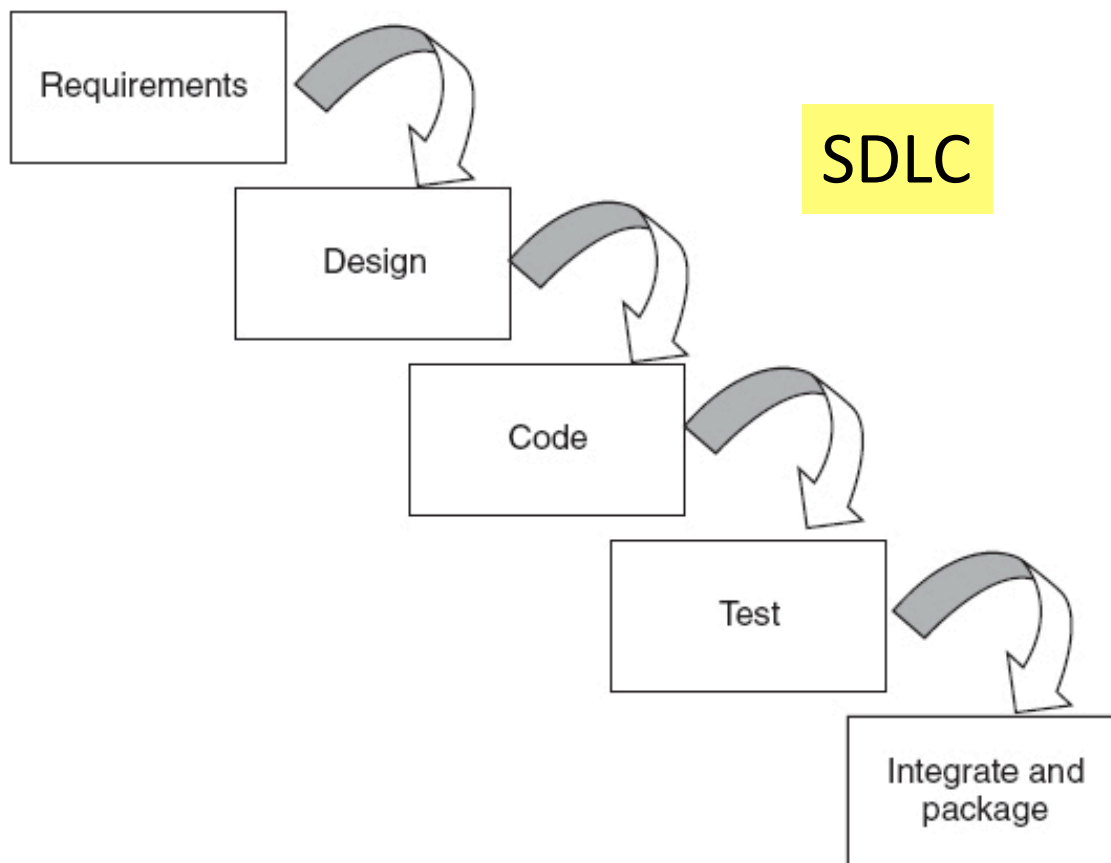
# Software Engineering

---

## Project Management

# Waterfall Model

Tsui & Bernal (2014), *Essentials of Software Engineering*



**Figure 4.2:** A waterfall model.



# Project Mgt

## MANAGE

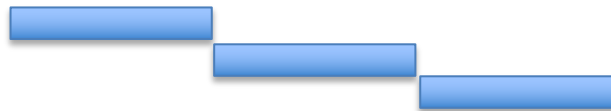
❖ Agile

❖ Scrum

❖ Waterfall model

- ◆ *Resources*
- ◆ *Time*
- ◆ *Cost*
- ◆ *Requirements*
- ◆ *Technologies*

Tasks



Time →

❖ Tools

- ☐ MS Project
- ☐ GANTT
- ☐ PERT

# Agile (PM/SE)

August 2018

Seventeen years ago **agile** began as a simple manifesto. Now, with all the methods and frameworks formulated in its name, it has become fat and flabby. We have reached a point where what we set out to change (big prescriptive methods) has returned, but now under the banner of being agile. The **Heart of Agile** is an attempt to return to agile's real core. But are the four words collaborate, deliver, reflect, and improve enough to get practitioners to implement the true heart of agile?

**Essence**, a new common ground for **software engineering** is an attempt to find a middle ground between the very core of agile and all the multitude of competing implementations of agile. In this presentation you will learn how Essence can strengthen the Heart of Agile without getting into particular ways of doing agile.

## ESSENCE | AN AGILE STANDARD

### ON COMMON GROUND

Essence – a standard that defines the smallest set of concepts that are common to all software projects – helps embed agile professional practices and governance across an organization for sustainable, scalable and responsive solution delivery.

The Essence standard helps teams navigate through many of the complex challenges common in software development delivery -from helping teams identify and engage with the right stakeholders at the right time in the right way, to making health and progress visible to all in a language that everyone can understand. It helps team move from software as a craft to engineering and puts them on a path of continuous learning.

# Essence

### ESSENCE AT GOOGLE

"Playing progress poker, reviewing health indicators and defining checkpoints triggered some good conversations in the team and helped crystallize some risks that he thought the team wasn't aware of and vice-versa." In order to explore Essence in a practical setting, a 2-day workshop was hosted at Google Switzerland on October 22, 2014.

**Martin Landers**

Engineering Manager, YouTube

- Measure health and progress visually in a method-agnostic way that everyone can understand.
- Easily determine where you and your team(s) are now and what's next.
- Small-scale endeavors can be smoothly scaled to larger-scale endeavors
- Large-scale approaches can be adapted in controlled and low-risk ways

# Section

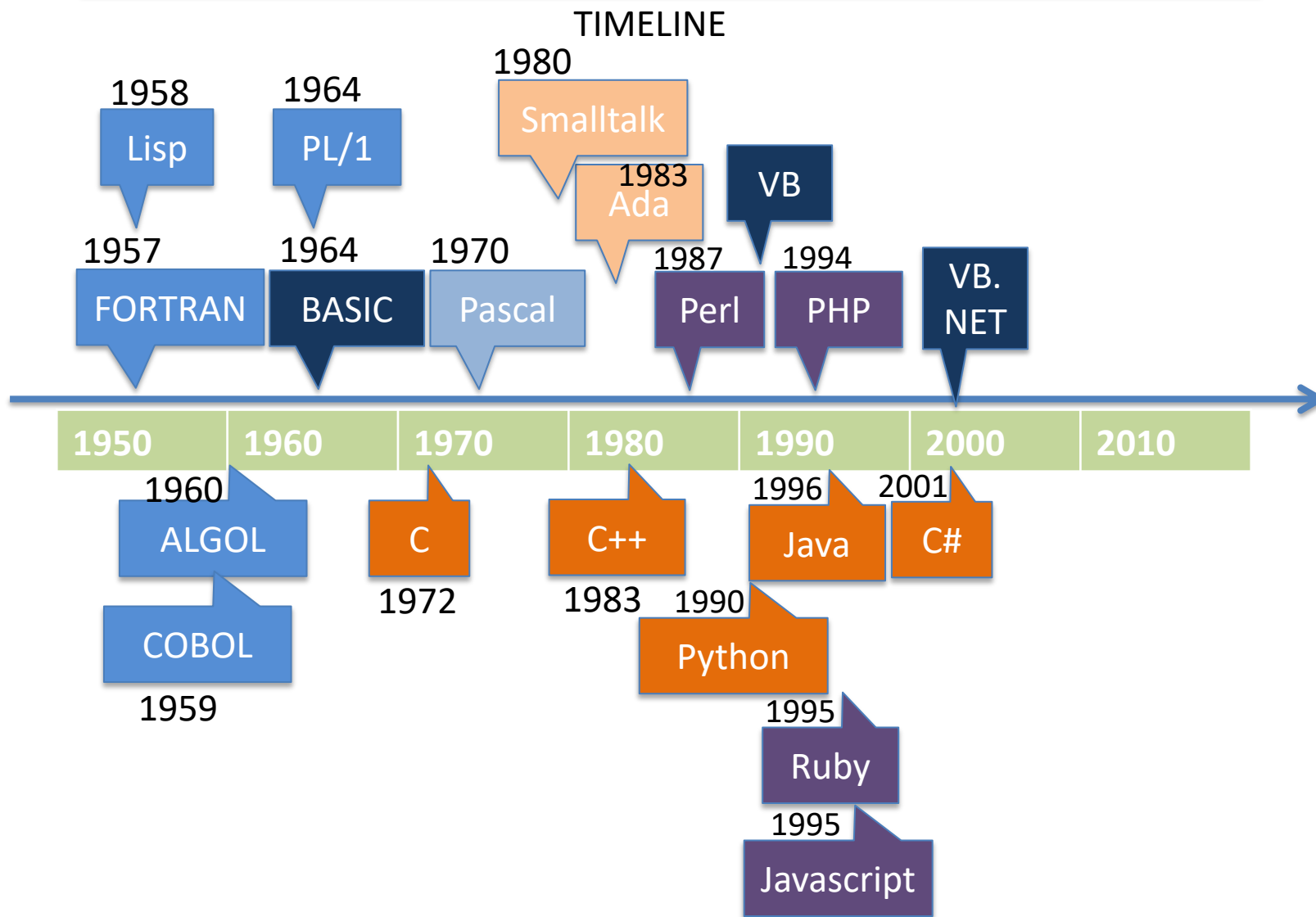
## High Level Languages

Java

Vs. Other HLLs



# HL Languages



# C History

"The Book"

© 1978

## THE C PROGRAMMING LANGUAGE

Brian W. Kernighan • Dennis M. Ritchie

PRENTICE-HALL SOFTWARE SERIES  
Brian W. Kernighan, Advisor

Copyright © 1978 by Bell Telephone Laboratories, Incorporated.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher. Printed in the United States of America. Published simultaneously in Canada.

This book was set in Times Roman and Courier 12 by the authors, using a Graphic Systems phototypesetter driven by a PDP-11/70 running under the UNIX operating system.

UNIX is a Trademark of Bell Laboratories.

➤ Kernighan & Ritchie



# C Design

C is a general-purpose programming language which features economy of expression, modern control flow and data structures, and a rich set of operators. C is not a “very high level” language, nor a “big” one, and is not specialized to any particular area of application. But its absence of restrictions and its generality make it more convenient and effective for many tasks than supposedly more powerful languages.

C was originally designed for and implemented on the UNIX† operating system on the DEC PDP-11, by Dennis Ritchie. The operating system, the C compiler, and essentially all UNIX applications programs (including all of the software used to prepare this book) are written in C. Production compilers also exist for several other machines, including the IBM System/370, the Honeywell 6000, and the Interdata 8/32. C is not tied to any particular hardware or system, however, and it is easy to write programs that will run without change on any machine that supports C.

C is a general-purpose programming language. It has been closely associated with the UNIX system since it was developed on that system, and since UNIX and its software are written in C. The language, however, is not tied to any one operating system or machine; and although it has been called a “system programming language” because it is useful for writing operating systems, it has been used equally well to write major numerical, text-processing, and data-base programs.

C is a relatively “low level” language. This characterization is not pejorative; it simply means that C deals with the same sort of objects that most computers do, namely characters, numbers, and addresses. These may be combined and moved about with the usual arithmetic and logical operators implemented by actual machines.



# C Contents

<b>Chapter 1</b>	<b>A Tutorial Introduction</b>
1.1	Getting Started
1.2	Variables and Arithmetic
1.3	The For Statement
1.4	Symbolic Constants
1.5	A Collection of Useful Programs
1.6	Arrays
1.7	Functions
1.8	Arguments — Call by Value
1.9	Character Arrays
1.10	Scope; External Variables
1.11	Summary

<b>Chapter 2</b>	<b>Types, Operators and Expressions</b>
2.1	Variable Names
2.2	Data Types and Sizes
2.3	Constants
2.4	Declarations
2.5	Arithmetic Operators
2.6	Relational and Logical Operators
2.7	<u>Type Conversions</u>
2.8	Increment and Decrement Operators
2.9	Bitwise Logical Operators
2.10	Assignment Operators and Expressions
2.11	Conditional Expressions
2.12	Precedence and Order of Evaluation

<b>Chapter 7</b>	<b>Input and Output</b>
7.1	Access to the Standard Library
7.2	Standard Input and Output — Getchar and Putchar
7.3	Formatted Output — <b>Printf</b>
7.4	Formatted Input — <b>Scanf</b>
7.5	In-memory Format Conversion
7.6	File Access
7.7	Error Handling — Stderr and Exit
7.8	Line Input and Output
7.9	Some Miscellaneous Functions

<b>Chapter 3</b>	<b>Control Flow</b>
3.1	Statements and Blocks
3.2	If-Else
3.3	Else-If
3.4	Switch
3.5	Loops — While and For
3.6	Loops — Do-while
3.7	Break
3.8	Continue
3.9	Goto's and Labels

<b>Chapter 4</b>	<b>Functions and Program Structure</b>
4.1	Basics
4.2	Functions Returning Non-Integers
4.3	More on Function Arguments
4.4	External Variables
4.5	Scope Rules
4.6	Static Variables
4.7	Register Variables
4.8	Block Structure
4.9	Initialization
4.10	Recursion
4.11	The C Preprocessor

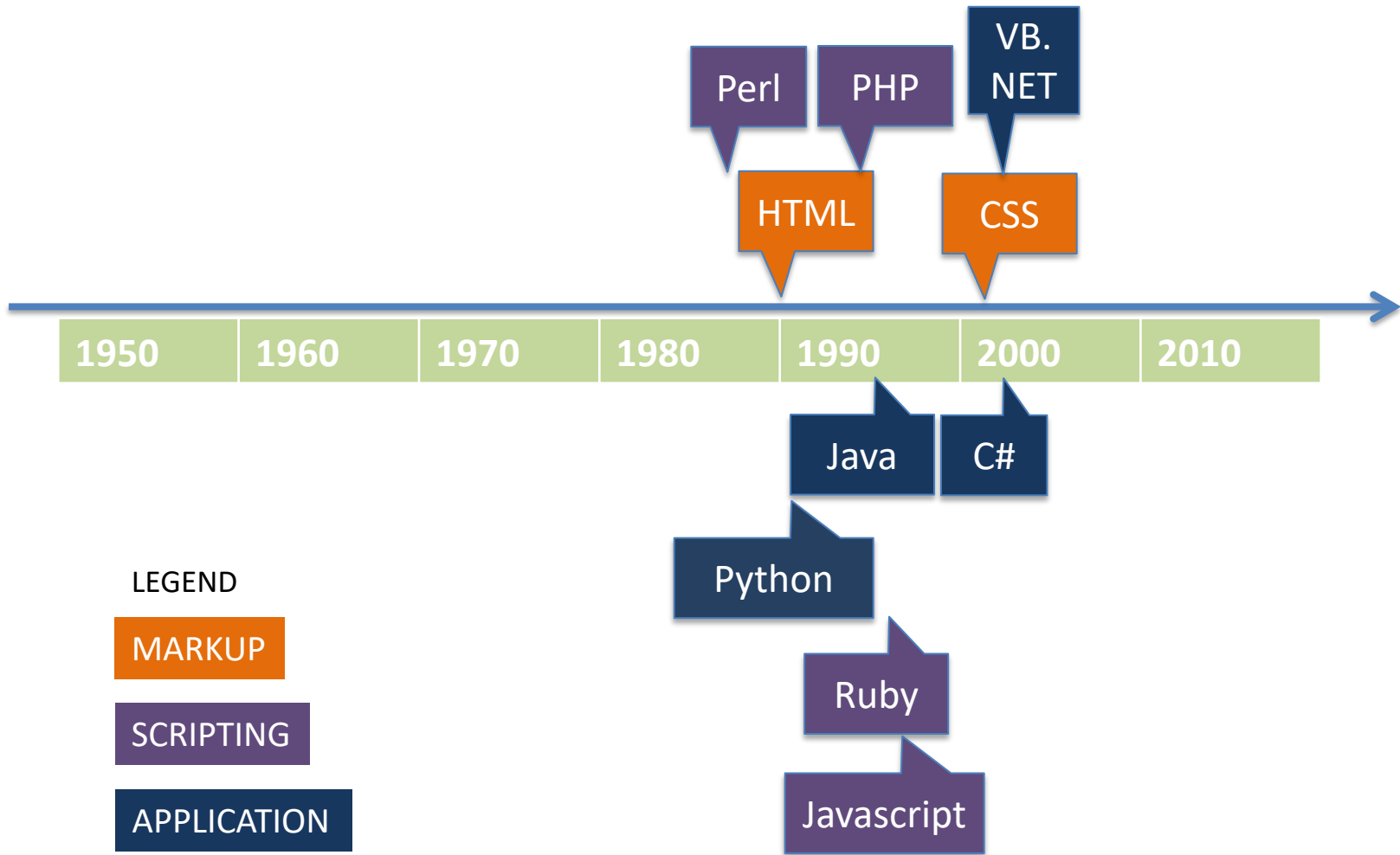
<b>Chapter 5</b>	<b>Pointers and Arrays</b>
5.1	Pointers and Addresses
5.2	Pointers and Function Arguments
5.3	Pointers and Arrays
5.4	Address Arithmetic
5.5	Character Pointers and Functions
5.6	Pointers are not Integers
5.7	Multi-Dimensional Arrays
5.8	Pointer Arrays; Pointers to Pointers
5.9	Initialization of Pointer Arrays
5.10	Pointers vs. Multi-dimensional Arrays
5.11	Command-line Arguments
5.12	Pointers to Functions

<b>Chapter 6</b>	<b>Structures</b>
6.1	Basics
6.2	Structures and Functions
6.3	Arrays of Structures

➤ **Pointers!**

# Web Languages

## TIMELINE



# HLL Popularity

Dec 2018

Table 1.7.1: Top languages ranked by popularity.

Language	Usage by percentage
Java	16%
C	14%
Python	8%
C++	8%
Visual Basic .NET	7%
C#	4%
JavaScript	3%
PHP	2%
SQL	2%
Objective-C	1%

(Source: <https://www.tiobe.com/tiobe-index/>)

# Key Languages & Tools



Languages: **Java**, C, C++, PHP, C#, R



Web: JavaScript, CSS3, HTML5, JQuery, Java EE, XML, Ajax, Github, JSP, Bootstrap

Databases: MySQL, Oracle DB

**SDLC** Agile, Scrum, Waterfall

Frameworks: Spring, Hibernate

Design Patterns: Factory, Singleton, Iterator, Builder, Observer, Command

Web/App Servers: Apache Tomcat, Xampp

Web Services: SOAP, REST



Tools: **Eclipse** Microsoft **Visual Studio**, Weka, SPSS, R Studio **jGRASP**

Platforms: Mac OS X, Windows, Linux (Ubuntu, Mint)

# Java Updates

## Security Concerns

Java SE 8 Update 101 [Citadel recommends removing or disabling Java from your browser. Java is a major source of cyber criminal exploits. It is not needed for most internet browsing. If you have a particular web site that requires Java, Citadel recommends using a two-browser approach to minimize risk. If you normally browse the Web with Firefox, for example, disable the Java plugin in Firefox and use an alternative browser — such as Chrome, IE9, Safari, etc — with Java enabled to browse only the sites that require it.]

## Java VM (not RTE) on Client

latest rev

SE 8 Update 181 (“SE 8u181”)

# High-Level Language *Types*

- ❖ Imperative/Procedural {main + subs}
  - ✧ **C**
  - ✧ FORTRAN, PL/1, Pascal (old ones)
- ❖ Object-Oriented (OOD/OOP) {classes + methods}
  - ✧ **C++, Java**
  - ✧ Visual Studio (.NET)-- **VB, C#**
  - ✧ Apple's Objective C/Swift
- ❖ Scripting (Functional) {command sequence}
  - ✧ Perl, Ruby, PHP, Javascript
- ❖ Markup {sequence independent page building}
  - ✧ HTML5, XML, XAML, CSS
- ❖ Database Query
  - ✧ SQL, MySQL, jQuery

# Web Server Side

## 8.1.3 Comparing Server-Side Technologies

### ❖ ASP

- ☐ Microsoft .NET
- ☐ JIT (intermediate language)
- ☐ MVC

### ❖ JSP

- ☐ Java
- ☐ JIT (intermediate language)

### ❖ Node.js

- ☐ Javascript (on server)
- ☐ Complete (app server too)

### ❖ Perl

- ☐ Scripting + C-like
- ☐ Uses CGI

### ❖ PHP

- ☐ Scripting + OOP
- ☐ JIT (intermediate language)

### ❖ Python

- ☐ Terse OOP
- ☐ Used in Django, Pyramid

### ❖ Ruby (on Rails)

- ☐ Templates
- ☐ MVC

### ❖ Python

- ☐ Terse OOP

# Software

---

Example:  
Hello World



# Comparison: “Hello World”

C

```
#include <stdio.h>
void main () {
    printf("Hello world!\n");
}
```

C++

```
#include <iostream>
void main () {
    std::cout << "Hello world!\n";
}
```

Java

```
Public class helloWorld {
public static void main (String[] args) {
    system.out.println ("Hello world!");
}
```

Javascript

```
//myfile.js
Console.log("Hello world!");
```

# Comparison: “Hello World”

Basic

```
10 PRINT "Hello, world!"  
20 END
```

note: line numbers!

VB

```
Public Sub Main()  
    MsgBox "Hello, world!"  
End Sub
```

OOP + **GUI**

C#

```
using System;  
  
internal static class HelloWorld  
{  
    private static void Main()  
    {  
        Console.WriteLine("Hello, world!");  
    }  
}
```

OOP + console

DOS

```
@echo Hello World!
```

script (for console)

# Comparison: “Hello World”

PHP

```
1 <?php
2 print "Hello world!";
3 ?>
```

➤ all console

Assembly

```
1 .model small
2 .stack 100h
3
4 .data
5 msg      db      'Hello world!$'
6
7 .code
8 start:
9         mov     ah, 09h
10        lea     dx, msg
11        int     21h
12        mov     ax, 4C00h ;
13        int     21h
14 end start
```

# "Hello World"-Java Applet

## Applet

*Main article: [Java applet](#)*

Java applets are programs that are embedded in other applications, typically in a Web page displayed in a web browser.

```
// Hello.java
import javax.swing.JApplet;
import java.awt.Graphics;

public class Hello extends JApplet {
    public void paintComponent(final Graphics g) {
        g.drawString("Hello, world!", 65, 95);
    }
}
```

"AWT" graphics  
has been replaced by  
"FX" graphics  
(along with "Swing")

The `import` statements direct the Java compiler to include the `javax.swing.JApplet` and `java.awt.Graphics` classes in the compilation. The import statement allows these classes to be referenced in the source code using the simple class name (i.e. `JApplet`) instead of the fully qualified class name (FQCN, i.e. `javax.swing.JApplet`).

The `Hello` class `extends` (subclasses) the `JApplet` (Java Applet) class; the `JApplet` class provides the framework for the host application to display and control the lifecycle of the applet. The `JApplet` class is a `JComponent` (Java Graphical Component) which provides the applet with the capability to display a graphical user interface (GUI) and respond to user events.

The `Hello` class overrides the `paintComponent(Graphics)` method (additionally indicated with the `Override` annotation, supported as of JDK 1.5, inherited from the `Container` superclass to provide the code to display the applet. The `paintComponent()` method is passed a `Graphics` object that contains the graphic context used to display the applet. The `paintComponent()` method calls the graphic context `drawString(String, int, int)` method to display the "Hello, world!" string at a pixel offset of (65, 95) from the upper-left corner in the applet's display.

# Compare Languages

---

C++ vs Java

# C, C++, C# vs. Java

## ❖ C (1971)

- origin: Base language – invented by Bell Labs for Unix
- level: low/system level (can embed assembly code)
- target: embedded systems (still most used)

## ❖ C++ (1979)

- origin: “C with classes” – invented by Bell Labs (Stroustrup)
- level/target: low/system/embedded + desktop (not web)
- flexible: supports “C-like” and/or OOP (classes) styles
- evolution: moving away from classes (OOP) → a “better C”

## ❖ C# (2003)

- origin: invented by Microsoft for its “.NET” apps on Windows only
- target: all applications, but mostly Web apps

## ❖ Java (2003)

- origin: invented by Sun Microsystems for machine independence
- target: all “client” computers on the WWW to run web apps
- evolution: now Oracle, has “desktop” and “enterprise” versions (EE)
- requires “JVM” or “JRE” to run programs

# "C" Application Languages

	C	C++	Java
Object-oriented	NO	MIXED	PURE
Compiled/Int	Compiled	Compiled	<i>Interpreted</i>
Intermediate Lang	none	none	<i>Bytecode</i>
Output	printf	std::cout <<	system.out.println
Input	scanf	std::cin >>	input.next( )
Strong Typing	NO	NO	YES
Source File.ext	.c	.cpp	.java

JVM/JRE

# Google: Java vs. C++

## INTRO

### What is it like to be a Java programmer at Google?



Jason Roselander, Java developer

Written Dec 31, 2013 · Upvoted by Jeff Nelson, Invented Chromebook, Former Googler  
and Miguel Paraz, professional Java programmer since 2002

I've only been here 6 months, working on 1 team the entire time, but I have a few impressions of Java at Google.

The main one is that C++ remains the primary programming language at Google. Google has developed an extremely sophisticated, reliable & efficient C++ codebase and they're not about to rewrite it in Java. Another reason seems to be that Google values memory & CPU utilization very highly, and C++ allows them to squeeze every last cycle and byte out of their hardware.

So Google engineers, especially ones who've been there a long time, are very proficient in C++. As a result, their Java tends to look a bit like C++. Massive source files which declare many inner classes are prevalent. Output parameters to methods are not uncommon. Exceptions are banned in Google C++ but not in Java, so exceptions aren't always used in the most idiomatic way.

As for tools & libraries, arguably two of the biggest libraries for Java outside of the Apache Commons have come out of Google: the Guava libraries and Guice. I used both of them at my previous job and assumed they'd be ubiquitous at Google. |



# C++ at Google

## Why did Google move from Python to C++ for use in its crawler?



Harald Tveit Alvestrand

Written Apr 20, 2015

When I had my first big project at Google, we wrote a proof-of-concept in Python, and then discussed language for the production version.

Performance was not an issue; correctness was.

Jeff Dean (yes, himself) recommended C++ - his words were something like "Python is maintainable until the guy who initially wrote the code leaves".

I still write stuff in Python, but sometimes I wonder if he spoke wisdom.

(we did switch to C++, and the project eventually failed. I don't think the language switch was relevant to the end result.)

121.5k Views · View Upvotes

# C++ vs. Java – Classes

## C++

```
class Foo {           // Declares class Foo
    int x;           // Private Member variable
public:
    Foo() : x(0)      // Constructor for Foo;
    initializes
    {}               // x to 0. If the initializer
                    // were
                    // omitted, the variable would
                    // not
                    // be initialized to a specific
                    // value.

    int bar(int i) { // Member function bar()
        return 3*i + x;
    }
};
```

```
Foo a;
// declares a to be a Foo object value,
// initialized using the default constructor.

// Another constructor can be used as
Foo a(args);
// or (C++11):
Foo a{args};
```

## Java

```
class Foo {           // Defines class Foo
    private int x;     // Member variable, normally
                    // declared
                    // as private to enforce
                    // encapsulation
                    // initialized to 0 by default

    public Foo() {     // Constructor for Foo
    }                 // no-arg constructor supplied
                    // by default

    public int bar(int i) { // Member method bar()
        return 3*i + x;
    }
}
```

```
Foo a = new Foo();
// declares a to be a reference to a new Foo object
// initialized using the default constructor

// Another constructor can be used as
Foo a = new Foo(args);
```

# C++ vs. Java – Misc

## INTRO

C++	Java
Extends <b>C</b> with <b>object-oriented programming</b> and <b>generic programming</b> . C code can most properly be used.	Strongly influenced by C++/C syntax.
Compatible with <b>C</b> source code, except for a few <b>corner cases</b> .	Provides the <b>Java Native Interface</b> and recently <b>Java Native Access</b> as a way to directly call C/C++ code.
<b>Write once, compile anywhere</b> (WOCA).	<b>Write once, run anywhere/everywhere</b> (WORA/WORE).
Allows <b>procedural programming</b> , <b>functional programming</b> , <b>object-oriented programming</b> , <b>generic programming</b> , and <b>template metaprogramming</b> . Favors a mix of paradigms.	Allows <b>procedural programming</b> , <b>functional programming</b> (since Java 8) and <b>generic programming</b> (since Java 5), but strongly encourages the <b>object-oriented programming paradigm</b> . Includes support for creating <b>scripting languages</b> .
Runs as native executable machine code for the target <b>instruction set(s)</b> .	Runs on a <b>virtual machine</b> .
Provides object types and type names. Allows reflection via <b>run-time type information</b> (RTTI).	Is <b>reflective</b> , allowing metaprogramming and dynamic code generation at runtime.
Has multiple binary compatibility standards (commonly Microsoft (for MSVC compiler) and Itanium/GNU (for almost all other compilers)).	Has one binary compatibility standard, <b>cross-platform</b> for OS and compiler.
Optional automated <b>bounds checking</b> (e.g., the <code>at( )</code> method in <code>vector</code> and <code>string</code> containers).	All operations are required to be bound-checked by all compliant distributions of Java. <b>HotSpot</b> can remove bounds checking.
Native <b>unsigned arithmetic</b> support.	Native unsigned arithmetic unsupported. Java 8 changes some of this, but aspects are unclear. <sup>[1]</sup>
Standardized minimum limits for all numerical types, but the actual sizes are implementation-defined. Standardized types are available via the standard library <code>&lt;cstdint&gt;</code> .	Standardized limits and sizes of all primitive types on all platforms.
Pointers, references, and pass-by-value are supported for all types (primitive or user-defined).	All types (primitive types and reference types) are always passed by value. <sup>[2]</sup>
<b>Memory management</b> can be done <b>manually</b> via <code>new / delete</code> , automatically by scope, or by smart pointers. Supports deterministic destruction of objects. <b>Garbage collection</b> ABI standardized in C++11, though compilers are not required to implement garbage collection.	Automatic <b>garbage collection</b> . Supports a non-deterministic <code>finalize()</code> method which use is not recommended. <sup>[3]</sup>

# C++ vs. Java – More

## C++

## Java

Operator <a href="#">overloading</a> for most operators. Preserving meaning (semantics) is highly recommended.	Operators are not overridable. The language overrides + and += for the String class.
Single and <a href="#">Multiple inheritance</a> of classes, including virtual inheritance.	Single inheritance of classes. Supports multiple inheritance via the <a href="#">Interfaces</a> construct, which is equivalent to a C++ class composed of abstract methods.
Compile-time templates. Allows for <a href="#">Turing complete</a> meta-programming.	<a href="#">Generics</a> are used to achieve basic type-parametrization, but they do not translate from source code to byte code due to the use of <a href="#">type erasure</a> by the compiler.
Function pointers, function objects, lambdas (in <a href="#">C++11</a> ), and interfaces.	References to functions achieved via the <a href="#">reflection</a> API. OOP idioms using Interfaces, such as Adapter, Observer, and Listener are generally preferred over direct references to methods.
No standard inline documentation mechanism. Third-party software (e.g. <a href="#">Doxygen</a> ) exists.	Extensive <a href="#">Javadoc</a> documentation standard on all system classes and methods.
<code>const</code> keyword for defining immutable variables and member functions that do not change the object. Const-ness is propagated as a means to enforce, at compile-time, correctness of the code with respect to mutability of objects (see <a href="#">const-correctness</a> ).	<code>final</code> provides a version of <code>const</code> , equivalent to <code>type* const</code> pointers for objects and <code>const</code> for primitive types. Immutability of object members achieved via read-only interfaces and object encapsulation.
Supports the <code>goto</code> statement.	Supports labels with loops and statement blocks.
Source code can be written to be <a href="#">cross-platform</a> (can be compiled for <a href="#">Windows</a> , <a href="#">BSD</a> , <a href="#">Linux</a> , <a href="#">OS X</a> , <a href="#">Solaris</a> , etc., without modification) and written to use platform-specific features. Typically compiled into native machine code, must be recompiled for each target platform.	Compiled into byte code for the <a href="#">JVM</a> . Byte code is dependent on the Java platform, but is typically independent of <a href="#">operating system</a> specific features.



# C++ vs. Java – Libraries

## C++ Std C Lib

## Java

The [C++ Standard Library](#) was designed to have a limited scope and functions, but includes language support, diagnostics, general utilities, strings, locales, containers, algorithms, [iterators](#), numerics, input/output, random number generators, regular expression parsing, threading facilities, type traits (for static type introspection) and [Standard C Library](#). The [Boost library](#) offers more functions including network I/O.

A rich amount of third-party libraries exist for GUI and other functions like: [Adaptive Communication Environment](#) (ACE), [Crypto++](#), various [XMPP Instant Messaging](#) (IM) libraries,<sup>[4]</sup> [OpenLDAP](#), [Qt](#), [gtkmm](#).

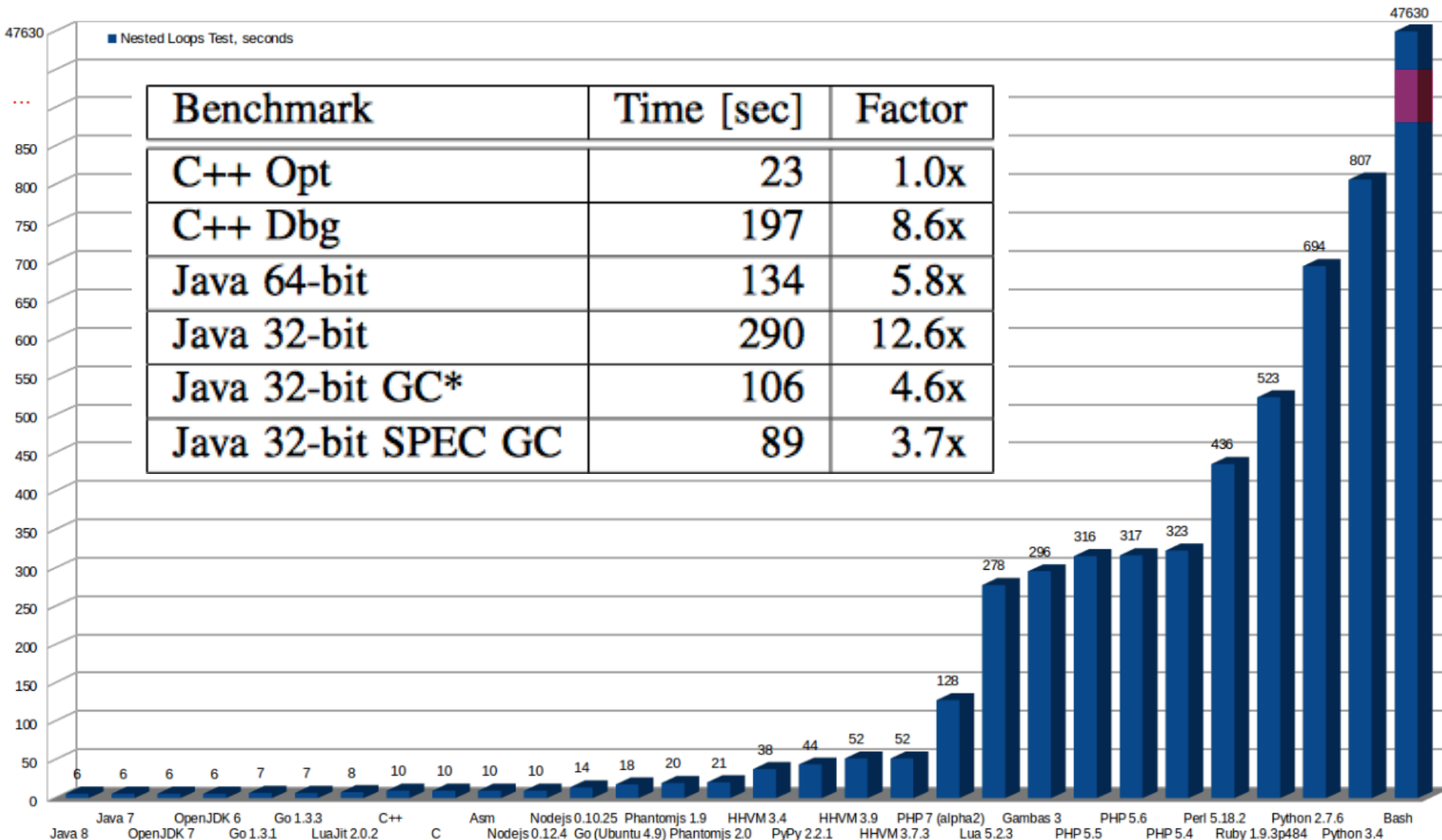
The standard library has grown with each release. By version 1.6, the library included support for locales, logging, containers and iterators, algorithms, GUI programming (but not using the system GUI), graphics, multi-threading, networking, platform security, introspection, dynamic class loading, blocking and non-blocking I/O. It provided interfaces or support classes for [XML](#), [XSLT](#), [MIDI](#), database connectivity, naming services (e.g. [LDAP](#)), cryptography, security services (e.g. [Kerberos](#)), print services, and web services. SWT offers an abstraction for platform-specific GUIs.

- The core libraries, which include:

## Java

- IO/NIO
- Networking
- Reflection
- Concurrency
- Generics
- Scripting/Compiler
- Functional Programming (Lambda, Streaming)
- Collection libraries that implement data structures such as lists, dictionaries, trees, sets, queues and double-ended queue, or stacks
- XML Processing (Parsing, Transforming, Validating) libraries
- Security<sup>[63]</sup>
- Internationalization and localization libraries<sup>[64]</sup>

# Speed Test

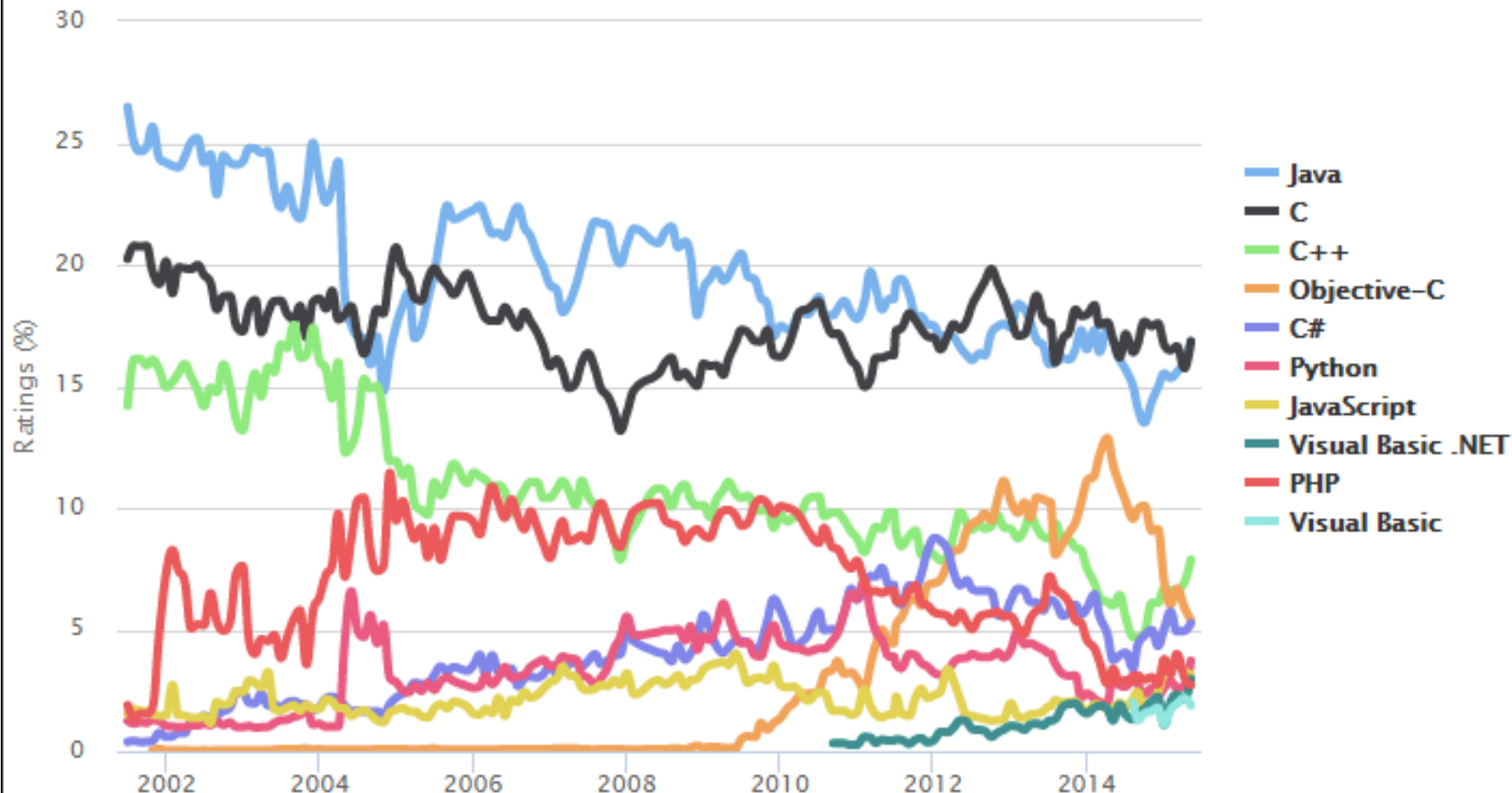


# HLL Comparison

## TIOBE Programming Community Index

Usage

Source: [www.tiobe.com](http://www.tiobe.com)



# HLL Comparison

Language ↕	Statements ratio <sup>[33]</sup> ↕	Lines ratio <sup>[34]</sup> ↕
C	1	1
C++	2.5	1
Fortran	2	0.8
Java	2.5	1.5
Perl	6	6
Smalltalk	6	6.25
Python	6	6.5

Code size



# HLL Comparison

Language ⇄	Intended use ⇄	Imperative ⇄	Object-oriented ⇄	Functional ⇄	Procedural ⇄	Generic ⇄	Reflective ⇄	Event-driven ⇄	Other paradigm(s) ⇄	Standardized? ⇄
<b>ActionScript 3.0</b>	Application, client-side, web	Yes	Yes					Yes		1996, ECMA
<b>Ada</b>	Application, embedded, realtime, <a href="#">system</a>	Yes	Yes <sup>[2]</sup>		Yes <sup>[3]</sup>	Yes <sup>[4]</sup>			concurrent, <sup>[5]</sup> distributed, <sup>[6]</sup>	1983, 2005, 2012, ANSI, ISO, <a href="#">GOST</a> 27831-88 <sup>[7]</sup>
<b>Aldor</b>	Highly domain-specific, symbolic computing	Yes	Yes	Yes						No
<b>ALGOL 58</b>	Application	Yes								No
<b>ALGOL 60</b>	Application	Yes								1960, <a href="#">IFIP WG 2.1</a> , ISO <sup>[8]</sup>
<b>ALGOL 68</b>	Application	Yes							concurrent	1968, <a href="#">IFIP WG 2.1</a> , <a href="#">GOST</a> 27974-88, <sup>[9]</sup>
<b>Ateji PX</b>	Parallel application		Yes						<a href="#">pi calculus</a>	No
<b>APL</b>	Application, data processing								array-oriented, tacit	1989, ISO
<b>Assembly language</b>	General	Yes							any, syntax is usually highly specific, related to the target processor	No
<b>BASIC</b>	Application, education	Yes			Yes					1983, <a href="#">ANSI</a> <a href="#">✓</a> , ISO


# HLL Comparison

## INTRO

Language ⇄	Intended use ⇄	Imperative ⇄	Object-oriented ⇄	Functional ⇄	Procedural ⇄	Generic ⇄	Reflective ⇄	Event-driven ⇄	Other paradigm(s) ⇄	Standardized? ⇄
<b>C</b>	Application, system, <sup>[11]</sup> general purpose, low-level operations	Yes			Yes					1989, ANSI C89, ISO C90, ISO C99, ISO C11 <sup>[12]</sup>
<b>C++</b>	Application, system	Yes	Yes	Yes	Yes	Yes				1998, ISO/IEC 1998, ISO/IEC 2003, ISO/IEC 2011, ISO/IEC 2014 <sup>[13]</sup>
<b>C#</b>	Application, RAD, business, client-side, general, server-side, web	Yes	Yes	Yes <sup>[14]</sup>	Yes	Yes	Yes	Yes	structured, concurrent	2000, ECMA, ISO <sup>[15]</sup>
<b>Go</b>	Application, web, server-side	Yes		Yes	Yes		Yes	Yes	concurrent	<i>De facto</i> standard via <a href="#">Go Language Specification</a> ↗
<b>Haskell</b>	Application			Yes		Yes			lazy evaluation	2010, Haskell 2010 <sup>[23]</sup>
<b>Java</b>	Application, business, client-side, general, mobile development, server-side, web	Yes	Yes	Yes	Yes	Yes	Yes	Yes	concurrent	<i>De facto</i> standard via <a href="#">Java Language Specification</a> ↗
<b>JavaScript</b>	Client-side, server-side, web	Yes	Yes	Yes			Yes		prototype-based	1997, ECMA

# HLL Comparison

## INTRO

Language ⇄	Intended use ⇄	Imperative ⇄	Object-oriented ⇄	Functional ⇄	Procedural ⇄	Generic ⇄	Reflective ⇄	Event-driven ⇄	Other paradigm(s) ⇄	Standardized? ⇄
<b>Pascal</b>	Application, education	Yes			Yes					1983, ISO <sup>[28]</sup>
<b>Perl</b>	Application, scripting, text processing, Web	Yes	Yes	Yes	Yes	Yes	Yes			No
<b>PHP</b>	Server-side, web application, web	Yes	Yes <sup>[29]</sup>	Yes <sup>[30]</sup>	Yes		Yes			No
<b>PL/I</b>	Application	Yes	Yes		Yes					1969
<b>Python</b>	Application, general, web, scripting, artificial intelligence, scientific computing	Yes	Yes	Yes	Yes		Yes		aspect-oriented	No
<b>Ruby</b>	Application, scripting, web	Yes	Yes	Yes			Yes		aspect-oriented	2011(JIS X 3017), 2012(ISO/IEC 30170)
<b>Scala</b>	Application, distributed, web	Yes	Yes	Yes		Yes	Yes	Yes		<i>De facto</i> standard via <a href="#">Scala Language Specification (SLS)</a> 
<b>Swift</b>	Application, general	Yes	Yes	Yes		Yes	Yes	Yes	concurrent	No
<b>Visual Basic .NET</b>	Application, RAD, education, web, business, general	Yes	Yes	Yes	Yes	Yes	Yes	Yes	structured, concurrent	No

# HLL Usage

## INTRO

Programming languages used in most popular websites\*

Websites	Popularity (unique visitors per month) <sup>[1]</sup>	Front- end (Client- side)	Back-end (Server-side)	Database	Notes
WordPress.com	240,000,000	JavaScript	PHP, JavaScript <sup>[27]</sup> (Node.js)	MySQL	
Wikipedia.org	475,000,000	JavaScript	PHP, Hack	MySQL <sup>[citation needed]</sup> , MariaDB <sup>[18]</sup>	"MediaWiki" is programmed in PHP, runs on HHVM; free online encyclopedia
Yahoo	750,000,000	JavaScript	JavaScript, <sup>[14]</sup> PHP	MySQL, PostgreSQL <sup>[15]</sup>	Yahoo is presently <sup>[when?]</sup> transitioning to Node.js <sup>[14]</sup>
eBay.com	285,000,000	JavaScript	Java, <sup>[21]</sup> JavaScript <sup>[22]</sup>	Oracle Database	Online auction house
LinkedIn.com	260,000,000	JavaScript	Java, JavaScript, <sup>[23]</sup> Scala	Voltdomort <sup>[24]</sup>	World's largest professional network
Amazon.com	500,000,000	JavaScript	Java, C++, Perl <sup>[16]</sup>	Oracle Database <sup>[17]</sup>	Popular internet shopping site
Facebook.com	900,000,000	JavaScript	Hack, PHP (HHVM), Python, C++, Java, Erlang, D, <sup>[9]</sup> Xhp, <sup>[10]</sup> Haskell <sup>[11]</sup>	MySQL, <sup>[12]</sup> HBase Cassandra <sup>[13]</sup>	The most visited social networking site
Pinterest	250,000,000	JavaScript	Django <sup>[25]</sup> (a Python framework), Erlang	MySQL, Redis <sup>[26]</sup>	
YouTube.com	1,100,000,000	JavaScript	C/C++, Python, Java, <sup>[6]</sup> Go <sup>[7]</sup>	BigTable, MariaDB <sup>[5][8]</sup>	The most visited video sharing site
Google.com <sup>[2]</sup>	1,200,000,000	JavaScript	C, C++, Go, <sup>[3]</sup> Java, Python	BigTable, <sup>[4]</sup> MariaDB <sup>[5]</sup>	The most used search engine in the world
Twitter.com	290,000,000	JavaScript	C++, Java, Scala, Ruby on Rails <sup>[19]</sup>	MySQL <sup>[20]</sup>	140 characters social network
Bing	285,000,000	JavaScript	ASP.NET	Microsoft SQL Server	
MSN.com	280,000,000	JavaScript	ASP.NET	Microsoft SQL Server	An email client, for simple use. Mostly known as "messenger".
Microsoft	270,000,000	JavaScript	ASP.NET	Microsoft SQL Server	Software company

# HLL Usage

Back-end (Server-side) table in most popular websites

Websites ⇅	ASP.NET ⇅	C ⇅	C++ ⇅	D ⇅	Erlang ⇅	Go ⇅	Hack ⇅	Java ⇅	JavaScript ⇅	Perl ⇅	PHP ⇅	Python ⇅	Ruby ⇅	Scala ⇅	Xhp ⇅
Google.com	No	Yes	Yes	No	No	Yes	No	Yes	No	No	No	Yes	No	No	No
YouTube.com	No	Yes	Yes	No	No	Yes	No	Yes	No	No	No	Yes	No	No	No
Facebook.com	No	No	Yes	Yes	Yes	No	Yes	Yes	No	No	Yes	Yes	No	No	Yes
Yahoo	No	No	No	No	No	No	No	No	Yes	No	Yes	No	No	No	No
Amazon.com	No	No	Yes	No	No	No	No	Yes	No	Yes	No	No	No	No	No
Wikipedia.org	No	No	No	No	No	No	No	No	No	No	Yes	No	No	No	No
Twitter.com	No	No	Yes	No	No	No	No	Yes	No	No	No	No	Yes	Yes	No
Bing	Yes	No	No	No	No	No	No	No	No	No	No	No	No	No	No
eBay.com	No	No	No	No	No	No	No	Yes	Yes	No	No	No	No	No	No
MSN.com	Yes	No	No	No	No	No	No	No	No	No	No	No	No	No	No
Microsoft															
Linkedin.com	No	No	No	No	No	No	No	Yes	Yes	No	No	No	No	Yes	No
Pinterest												Yes			
Ask.com															
WordPress.com	No	No	No	No	No	No	No	No	Yes	No	Yes	No	No	No	No

# HLL-Java

Java

# Java Intro



<b>Paradigm</b>	Multi-paradigm: Object-oriented (class-based), structured, imperative, generic, reflective, concurrent
<b>Designed by</b>	James Gosling
<b>Developer</b>	Sun Microsystems (now owned by Oracle Corporation)
<b>First appeared</b>	May 23, 1995; 21 years ago <sup>[1]</sup>
<b>Typing discipline</b>	Static, strong, safe, nominative, manifest
<b>License</b>	GNU General Public License, Java Community Process
<b>Filename extensions</b>	.java , .class, .jar
<b>Website</b>	<a href="http://java.net">java.net</a>

## Major implementations

OpenJDK, GNU Compiler for Java (GCJ), many others

## Dialects

Generic Java, Pizza

## Principles

There were five primary goals in the creation of the Java language:<sup>[16]</sup>

1. It must be "simple, object-oriented, and familiar".
2. It must be "robust and secure".
3. It must be "architecture-neutral and portable".
4. It must execute with "high performance".
5. It must be "interpreted, threaded, and dynamic".

## Versions

*Main article: [Java version history](#)*

As of 2015, only Java 8 is supported ("publicly"). Major releases:

- JDK 1.0 (January 21, 1996)
- JDK 1.1 (February 19, 1997)
- J2SE 1.2 (December 8, 1998)
- J2SE 1.3 (May 8, 2000)
- J2SE 1.4 (February 6, 2002)
- J2SE 5.0 (September 30, 2004)
- Java SE 6 (December 11, 2006)
- Java SE 7 (July 28, 2011)
- Java SE 8 (March 18, 2014)

<b>Designed by</b>	James Gosling
<b>Developer</b>	Sun Microsystems (now owned by Oracle Corporation)
<b>First appeared</b>	May 23, 1995; 21 years ago <sup>[1]</sup>
<b>Typing discipline</b>	Static, strong, safe, nominative, manifest
<b>License</b>	GNU General Public License, Java Community Process
<b>Filename extensions</b>	.java , .class, .jar
<b>Website</b>	<a href="http://java.net">java.net</a>

## Major implementations

OpenJDK, GNU Compiler for Java (GCJ), many others

## Dialects

Generic Java, Pizza

## Influenced by

Ada 83, C++,<sup>[2]</sup> C#,<sup>[3]</sup> Eiffel,<sup>[4]</sup> Generic Java, Mesa,<sup>[5]</sup> Modula-3,<sup>[6]</sup> Oberon,<sup>[7]</sup> Objective-C,<sup>[8]</sup> UCSD Pascal,<sup>[9][10]</sup> Object Pascal<sup>[11]</sup>

## Influenced

Ada 2005, BeanShell, C#, Chapel,<sup>[12]</sup> Clojure, ECMAScript, Fantom, Groovy, Hack,<sup>[13]</sup> Haxe, J#, JavaScript, Kotlin, PHP, Python, Scala, Seed7, Vala



# Java History

## INTRO

### History Wikipedia

See also: [Java \(software platform\) § History](#)

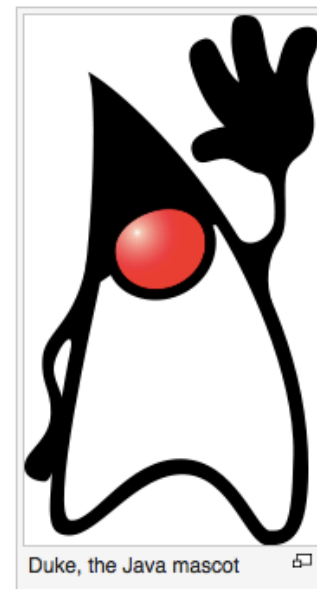
[James Gosling](#), Mike Sheridan, and [Patrick Naughton](#) initiated the Java language project in June 1991.<sup>[22]</sup> Java was originally designed for interactive television, but it was too advanced for the digital cable television industry at the time.<sup>[23]</sup> The language was initially called *Oak* after an *oak* tree that stood outside Gosling's office. Later the project went by the name *Green* and was finally renamed *Java*, from *Java coffee*.<sup>[24]</sup> Gosling designed Java with a C/C++-style syntax that system and application programmers would find familiar.<sup>[25]</sup>

[Sun Microsystems](#) released the first public implementation as Java 1.0 in 1995.<sup>[26]</sup> It promised "Write Once, Run Anywhere" (WORA), providing no-cost run-times on popular [platforms](#). Fairly secure and featuring configurable security, it allowed network- and file-access restrictions. Major [web browsers](#) soon incorporated the ability to run *Java applets* within web pages, and Java quickly became popular, while mostly outside of browsers, that wasn't the original plan. In January 2016, Oracle announced that Java runtime environments based on JDK 9 will discontinue the browser plugin.<sup>[27]</sup> The Java 1.0 compiler was re-written in Java by [Arthur van Hoff](#) to comply strictly with the Java 1.0 language specification.<sup>[28]</sup> With the advent of *Java 2* (released initially as J2SE 1.2 in December 1998 – 1999), new versions had multiple configurations built for different types of platforms. *J2EE* included technologies and APIs for enterprise applications typically run in server environments, while *J2ME* featured APIs optimized for mobile applications. The desktop version was renamed *J2SE*. In 2006, for marketing purposes, Sun renamed new J2 versions as *Java EE*, *Java ME*, and *Java SE*, respectively.

In 1997, Sun Microsystems approached the [ISO/IEC JTC 1](#) standards body and later the [Ecma International](#) to formalize Java, but it soon withdrew from the process.<sup>[29][30][31]</sup> Java remains a *de facto standard*, controlled through the [Java Community Process](#).<sup>[32]</sup> At one time, Sun made most of its Java implementations available without charge, despite their [proprietary software](#) status. Sun generated revenue from Java through the selling of licenses for specialized products such as the Java Enterprise System.

On November 13, 2006, Sun released much of its Java virtual machine (JVM) as [free and open-source software](#), (FOSS), under the terms of the [GNU General Public License](#) (GPL). On May 8, 2007, Sun finished the process, making all of its JVM's core code available under [free software](#)/open-source distribution terms, aside from a small portion of code to which Sun did not hold the copyright.<sup>[33]</sup>

Sun's vice-president Rich Green said that Sun's ideal role with regard to Java was as an "evangelist".<sup>[34]</sup> Following [Oracle Corporation](#)'s acquisition of Sun Microsystems in 2009–10, Oracle has described itself as the "steward of Java technology with a relentless commitment to fostering a community of participation and transparency".<sup>[35]</sup> This did not prevent Oracle from filing a lawsuit against Google shortly after that for using Java inside the Android SDK (see Google section below). Java



[James Gosling](#), the creator of Java (2008)



# Java Description

## Java (programming language)

From Wikipedia, the free encyclopedia

*"Java language" redirects here. For the natural language from the Indonesian island of Java, see [Javanese language](#). This article is about a programming language. For the software package downloaded from java.com, see [Java SE](#). Not to be confused with [JavaScript](#).*

**Java** is a general-purpose [computer programming language](#) that is [concurrent](#), [class-based](#), [object-oriented](#),<sup>[14]</sup> and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "[write once, run anywhere](#)" (WORA),<sup>[15]</sup> meaning that [compiled](#) Java code can run on all platforms that support Java without the need for recompilation.<sup>[16]</sup> Java applications are typically compiled to [bytecode](#) that can run on any [Java virtual machine](#) (JVM) regardless of [computer architecture](#). As of 2016, Java is one of the most [popular programming languages in use](#),<sup>[17][18][19][20]</sup> particularly for client-server web applications, with a reported 9 million developers.<sup>[21]</sup> Java was originally developed by [James Gosling](#) at [Sun Microsystems](#) (which has since been [acquired by Oracle Corporation](#)) and released in 1995 as a core component of Sun Microsystems' [Java platform](#). The language derives much of its [syntax](#) from [C](#) and [C++](#), but it has fewer [low-level](#) facilities than either of them.

The original and [reference implementation](#) Java [compilers](#), virtual machines, and [class libraries](#) were originally released by Sun under proprietary licences. As of May 2007, in compliance with the specifications of the [Java Community Process](#), Sun [relicensed](#) most of its Java technologies under the [GNU General Public License](#). Others have also developed alternative implementations of these Sun technologies, such as the [GNU Compiler for Java](#) (bytecode compiler), [GNU Classpath](#) (standard libraries), and [IcedTea-Web](#) (browser plugin for applets).

The latest version is Java 8, which is the only version currently supported for free by Oracle, although earlier versions are supported both by Oracle and other companies on a commercial basis.

# Java Versions

Jan 30, 2018



## A new version of Java is available!

Java 8 Update 161 build 12 is now available—you have Java 8 Update 151 build 12. Would you like to download it now?



Installing Java ...

ATMs, Smartcards, POS Terminals, Blu-ray Players, PCs  
Set Top Boxes, Modems, Servers, Switches  
Routers, Smartphones, Medical Devices  
Automobiles, Lottery Systems, Access Control Systems, Building Controls  
Programs, Modules...

# 3 Billion Devices Run Java

 **Java** | #1 Development Platform 



**You already have Java installed. To improve security and save energy, Safari did not activate Java after it was installed.**

When you visit a website that needs Java, you can turn it on in Safari Websites preferences.

Open Safari Websites Preferences

OK

# Java

## INTRO

Wikipedia

### Java platform

*Main articles: [Java \(software platform\)](#) and [Java virtual machine](#)*

❖ portability

One design goal of Java is portability, which means that programs written for the Java platform must run similarly on any combination of hardware and operating system with adequate runtime support. This is achieved by compiling the Java language code to an intermediate representation called [Java bytecode](#), instead of directly to architecture-specific [machine code](#). Java bytecode instructions are analogous to machine code, but they are intended to be executed by a [virtual machine](#) (VM) written specifically for the host hardware. [End users](#) commonly use a [Java Runtime Environment](#) (JRE) installed on their own machine for standalone Java applications, or in a web browser for Java [applets](#).

❖ JRE

Standard libraries provide a generic way to access host-specific features such as graphics, [threading](#), and [networking](#).

The use of universal bytecode makes porting simple. However, the overhead of interpreting bytecode into machine instructions makes interpreted programs almost always run more slowly than native [executables](#). However, [just-in-time](#) (JIT) compilers that compile bytecodes to machine code during runtime were introduced from an early stage. Java itself is platform-independent, and is adapted to the particular platform it is to run on by a [Java virtual machine](#) for it, which translates the [Java bytecode](#) into the platform's machine language.<sup>[38]</sup>

❖ Interpreter

❖ JIT



# Java Syntax & Source Files

## Syntax

*Main article: [Java syntax](#)*

The syntax of Java is largely influenced by [C++](#). Unlike C++, which combines the syntax for structured, generic, and object-oriented programming, Java was built almost exclusively as an object-oriented language.<sup>[16]</sup> All code is written inside classes, and every data item is an object, with the exception of the primitive data types, *i.e.* integers, floating-point numbers, [boolean values](#), and characters, which are not objects for performance reasons. Java reuses some popular aspects of C++ (such as `printf()` method).

Unlike C++, Java does not support [operator overloading](#)<sup>[47]</sup> or [multiple inheritance](#) for *classes*, though multiple inheritance is supported for [interfaces](#).<sup>[48]</sup> This simplifies the language and aids in preventing potential errors and [anti-pattern](#) design.<sup>[citation needed]</sup>

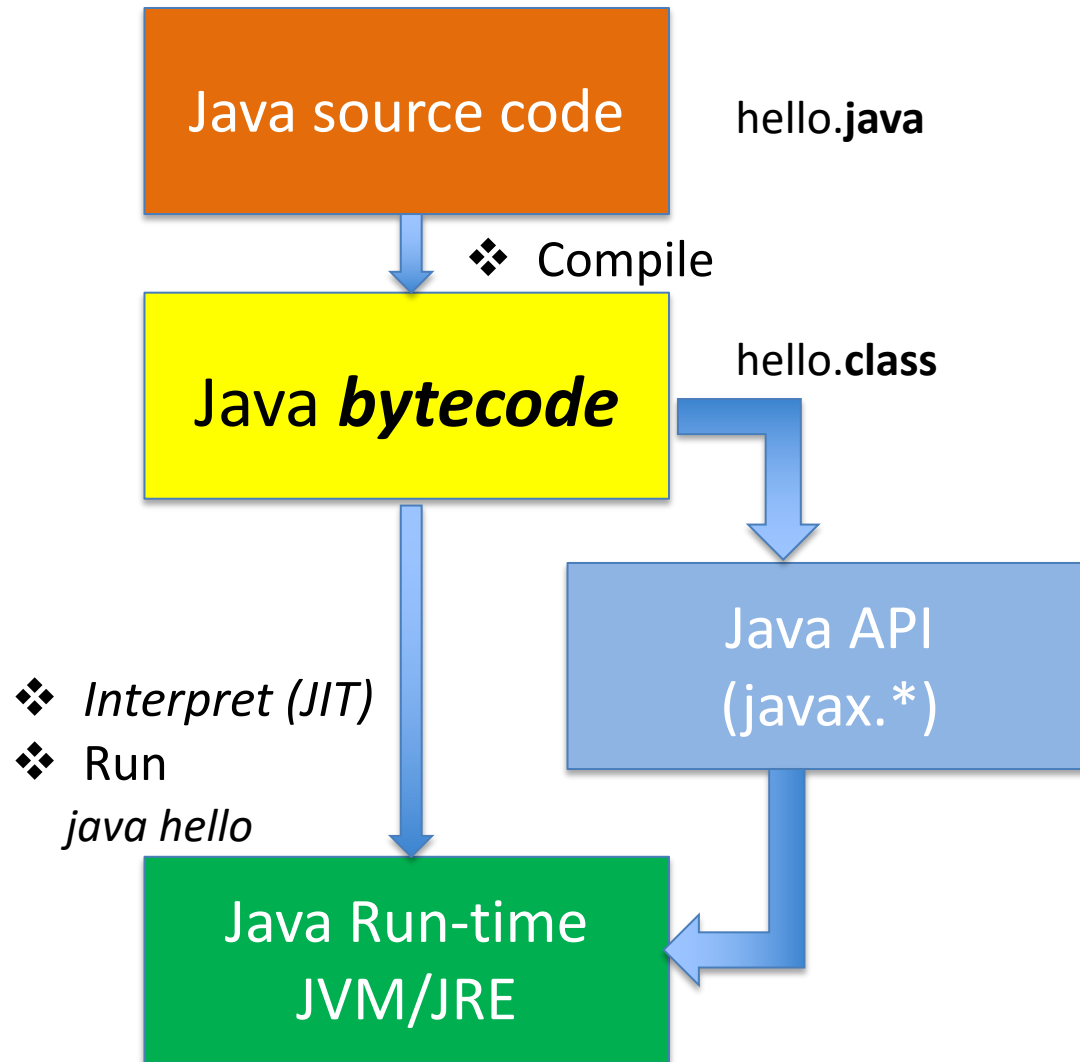
Java uses comments similar to those of C++. There are three different styles of comments: a single line style marked with two slashes (`//`), a multiple line style opened with `/*` and closed with `*/`, and the [Javadoc](#) commenting style opened with `/**` and closed with `*/`. The Javadoc style of commenting allows the user to run the Javadoc executable to create documentation for the program.

## Java Source Files

Source files must be named after the public class they contain, appending the suffix `.java`, for example, `HelloWorldApp.java`. It must first be compiled into bytecode, using a [Java compiler](#), producing a file named `HelloWorldApp.class`. Only then can it be executed, or "launched". The Java source file may only contain one public class, but it can contain multiple classes with other than public access and any number of public [inner classes](#). When the source file contains multiple classes, make one class "public" and name the source file with that public class name.

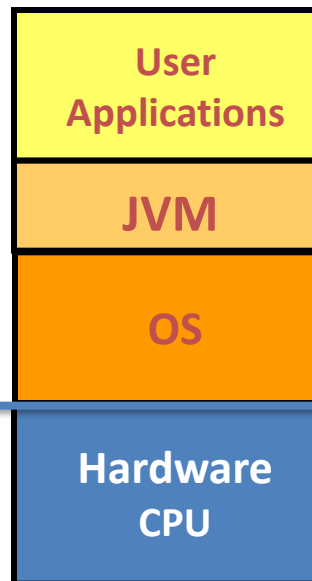
A `class` that is not declared `public` may be stored in any .java file. The compiler will generate a class file for each class defined in the source file. The name of the class file is the name of the class, with `.class` appended. For class file generation, [anonymous classes](#) are treated as if their name were the concatenation of the name of their enclosing class, a \$, and an integer.

# Running Java



# JVM

Portable language via Interpreter (JVM)

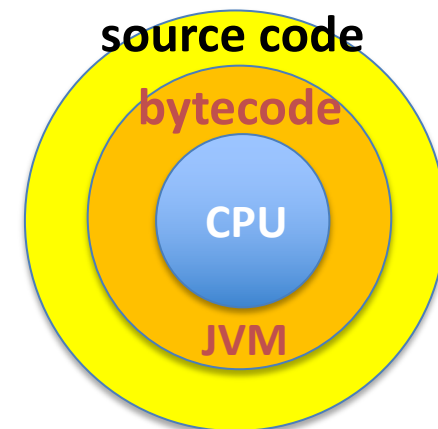


STACK MODEL

sourcecode.java

bytecode.class

- ❖ Windows
- ❖ Mac OS
- ❖ Unix



ONION SKIN MODEL

# Software

---

# Platforms



# Software Platforms

---

## ❖ *Standalone* Applications

- ☐ Native
  - Desktop
  - Mobil *apps* (phone/tablet)
- ☐ Web
  - Client ("Front end" via browser)
  - Server ("Back end")

## ❖ *Embedded Control*

- ☐ Appliances
- ☐ Cars/airplanes
- ☐ Phones/tablets
  - iOS
  - Android
- ☐ Computer Peripherals
  - Storage devices
  - Printers

# App Types

---

## ❖ NATIVE

- ✧ Runs directly on the device/computer on its OS
  - **Computer** (desktop or laptop)
  - **Mobile** (phone or tablet)

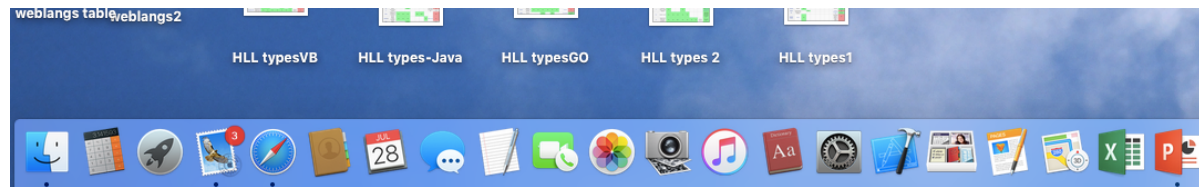
## ❖ WEB

- ✧ Runs remotely on the website server and is *displayed* on the device/computer via its **Browser**

## ❖ Mobile Web Apps

- ✧ redesigned websites for display on *mobile devices* (phones, tablets) that include applications (“Web Apps”)

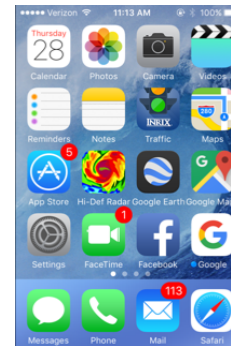
# Standalone Platforms



## ❖ Standalone Applications

### ❑ Native

- Desktop
  - Universal ("Office")
  - **Specialized <- THIS CLASS**
- Mobil *apps* (phone/tablet)





### ❑ Web

- Client ("Front end" via browser)
  - Desktop
  - Mobil
- Server ("Back end")





# Popular iPhone Apps

< Featured Best of 2015

Apps	Games
<p>1. App of the Year</p> <p>This game-changer made sharing and watching live videos an instant obsession.</p>  <p>Periscope Twitter, Inc.</p> <p>GET</p>	<p>1. Game of the Year</p> <p>Tomb Raider's puzzler spinoff floored us with its beauty and clever design.</p>  <p>Lara Croft GO SQUARE ENIX INC</p> <p>+ \$1.99</p> <p>In-App</p>

< Featured Best of 2015

<p>2. Runner-Up</p> <p>The best all-in-one photo editor provides powerful tools that are easy to use.</p>  <p>Enlight Lightricks Ltd.</p> <p>+ \$3.99</p>	<p>2. Runner-Up</p> <p>Managing life in Fallout's famous vaults is a standout strategy experience.</p>  <p>Fallout Shelter Bethesda</p> <p>+ GET</p> <p>In-App</p>
--	---

# Software

---

Tools  
SDK/IDE

## INTRO

### ❖ Compilers

- *Compiled* languages (C, C++, C#, VB)
  - ✧ Compile *completely*: Translate HLL (.c, .h) into ASM (.asm)
- *Interpreted* languages (**Java**, Pascal)
  - ✧ Compile *incompletely* (“JIT”) to an “intermediate” language
  - ✧ “Pseudo” code is compiled at run time (slow)

### ❖ Assemblers

- ✧ Translate ASM (.asm) into linkable machine code modules (“LM”)

### ❖ Linkers

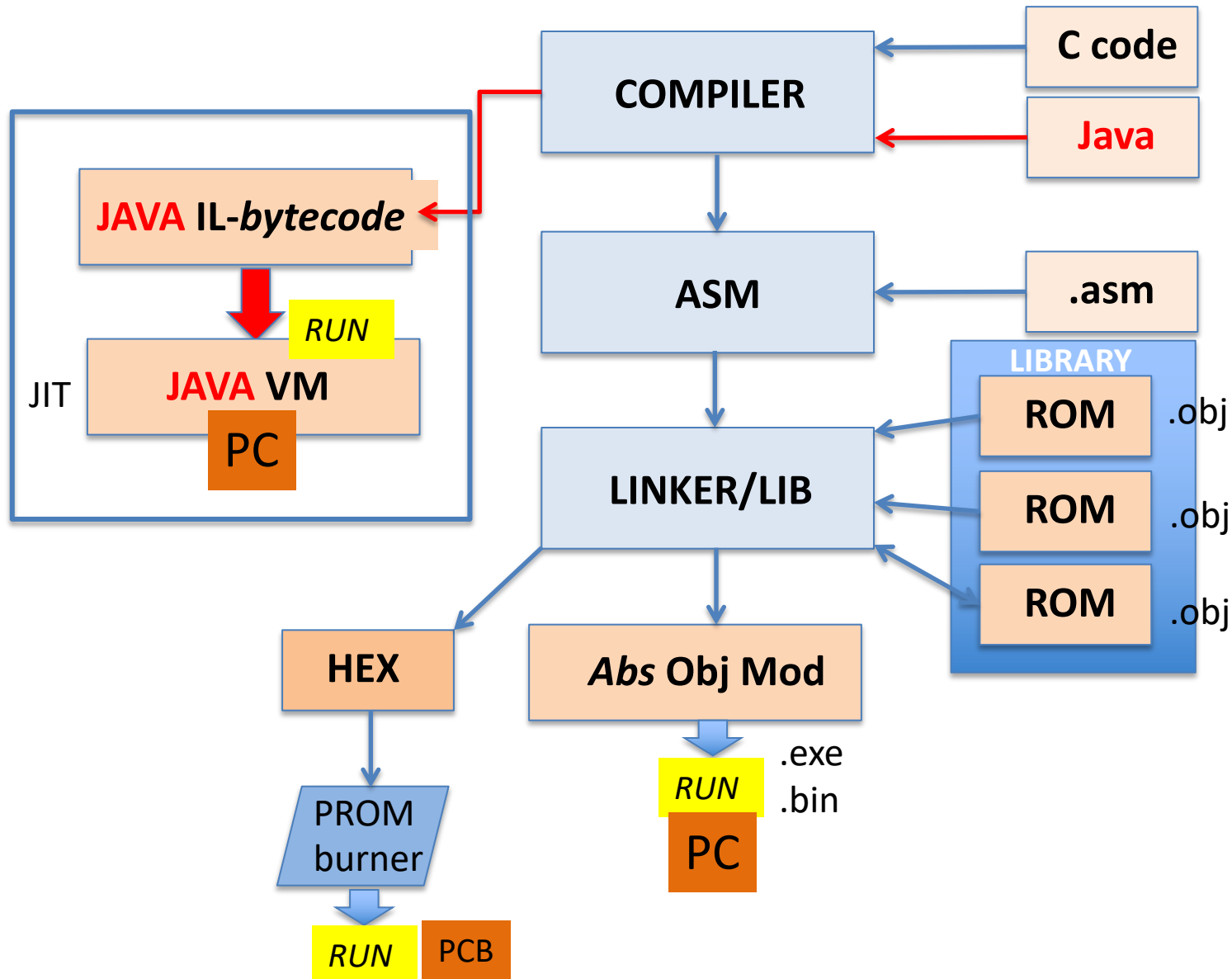
- ✧ Combine (“link”) LM modules into a single “executable” (.exe)
- ✧ Resolve external references
- ✧ Embed calls to dynamic “link libraries” or “frameworks” (.dll files)

### ❖ Debuggers

❖ *SDK contains Compilers + API (Libraries) + IDE*

❖ *IDE is a development environment w/debugger*

# IDE + Platforms





# SDK

❖ ***SDK = IDE & Compiler (for C/C++)***

- ☐ Eclipse
- ☐ GNU gcc
  - Windows
  - Pi (MinGW)
- ☐ MS Visual Studio
- ☐ IAR

# Running/Debugging


*RUN*

## Embedded Systems

### SIMULATOR

*SOFTWARE*

PC

- ☐ Debugger runtime environment in (IDE)
- ☐ Breakpoints
- ☐ Watch variables
- ☐ Target device selection
  - PC
  - Phone/tablet
  - *Board* 

### EMULATOR

*Substitute  
HARDWARE*

PCB

- ☐ ICE (in-circuit)
  - Pods
  - Breakpoints
  - Trace triggers & buffers
- ☐ Memory (known good)
  - R/W (ROM/RAM)
  - Wait states

### HARDWARE

*Actual  
HARDWARE*

PCB

- ☐ Code burned into ROM
- ☐ Working RAM
- ☐ Can use ICE
- ☐ Board bring-up
- ☐ Built-in test
  - JTAG
- ☐ Logic analyzers

# SDK/IDE

SOFTWARE DEV KIT  
INTEGRATED DEV ENVIRONMENT

## ❖ SDK

➤ **JDK**

## ❖ IDE

➤ **jGrasp**

➤ Eclipse

➤ MPLab

## ❖ SDK+IDE

➤ MS Visual Studio

- .NET
- UWP – Cross-platform

➤ Apple Xcode

# Development Platforms

## ❖ Design

### Software Applications: *Development Platforms*

#### ☐ Microsoft

- ✧ OS = Windows (7, 8, 10)
- ✧ API = .NET Framework
- ✧ SDK/IDE = **Visual Studio** *Cross Platform*
- ✧ Languages = .NET versions of VB, C#, C++, Java

#### ☐ Apple

- ✧ OS = Mac OS X, iOS (mobile)
- ✧ API = Xcode (Cocoa Touch)
- ✧ SDK/IDE = **Xcode**
- ✧ Languages = Objective C, Swift

#### ☐ Google

- ✧ OS = Android
- ✧ API = Android
- ✧ SDK/IDE = **Android**
- ✧ Languages = C++

# Win Dev – UWP

WINDOWS 10

## Universal Windows Platform

One concept behind the Universal Windows Platform is to make it as easy as possible to develop for Windows regardless of your primary platform...

- One binary / package
- One store
- Language of your choice
  - C#, VB, C++ with XAML
  - JavaScript with HTML/CSS
  - C++ with Microsoft DirectX
  - Android: Java/C++
  - iOS: Objective C (not Swift...yet)
- Android and iOS projects can be pulled into Visual Studio and compiled just like any other Windows application, creating the same single binary/package that will run on Windows devices just like native Windows applications



## C# and the Code Behind

- C# isn't the only language that can be used to write UWPA's, you can also use Visual Basic, C++, Java, JavaScript, and Objective C, but here we're focusing on C#
- In a Visual Studio solution, C# is contained in "code-behind/code-beside" files that are associated with XAML files, such as MainPage.xaml.cs, as well as standard code files like classes
- Each XAML file has a .xaml.cs code-behind file that contains all of the C# code that helps the XAML do stuff, like event handlers for UI elements, setting data contexts for UI element data bindings, and other things not easily done in XAML or more easily done in C#
- If you're just learning C#, this is the same C# you've been using to write console applications while learning basic concepts, the only difference is you're using a lot more of the API

### New Screencasts



Build a .NET MVC App

### New Course



#### Try .NET MVC

Learn the basics of building web applications with  
ASP.NET MVC.

Learn More

# MS Visual Studio



Community 2015

Choose your installation location

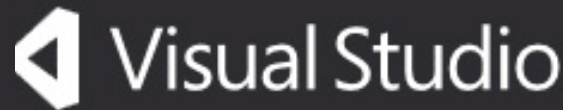
C:\Program Files (x86)\Microsoft Visual Studio 14.0

Setup requires up to 6 GB across all drives.

Choose the type of installation

- ☐ Typical  
Includes C#/VB Web and Desktop features
- ☒ Custom  
Allows you to customize features for your installation

You can add or remove additional features at any time after setup via Programs and Features in the Control Panel.



Community 2015

Select features

- ☒ Programming Languages
  - ☒ Visual C++
  - ☐ Visual F#
  - ☐ Python Tools for Visual Studio
- ☐ Windows and Web Development
- ☐ Cross Platform Mobile Development
- ☐ Common Tools



# iOS – Xcode

Objective C



Swift

## Xcode

Version 7.2 (7C68)

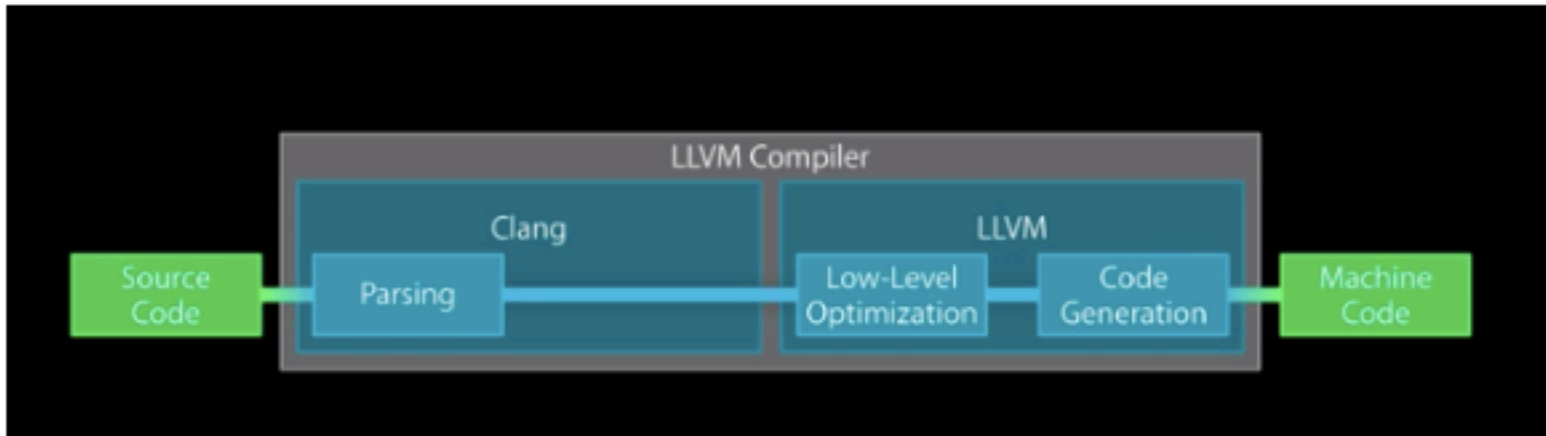
Copyright © 1999–2015 Apple Inc. All rights reserved. Apple and the Apple logo are trademarks of Apple Inc., registered in the U.S. and other countries.

[Acknowledgments](#)

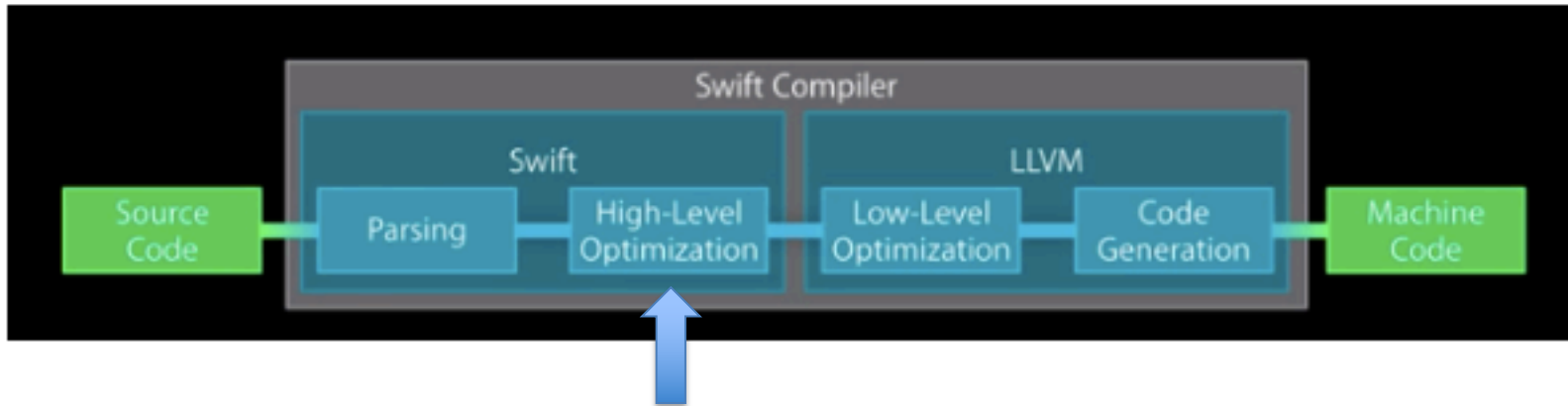
[License Agreement](#)

# iOS – Xcode

## Objective C



## Swift



# iOS – Xcode – Obj C

## Objective-C

Spaceship.h

```
#import "Vehicle.h"
#import "Planet.h"

@interface Spaceship : Vehicle
// declaration of public methods

@property (nonatomic) double topSpeed;

- (void)orbitPlanet:(Planet *)aPlanet
  atAltitude:(double)km;
```

Here's another example of sending a message.  
It looks like this method has 2 arguments:  
a Planet to travel to and a speed to travel at.  
It is being sent to an instance of Wormhole.

@end

Spaceship.m

```
#import "Spaceship.h"

@interface Spaceship()
// declaration of private methods (as needed)
@property (nonatomic, strong) Wormhole *nearestWormhole;
@end

@implementation Spaceship
// implementation of public and private methods

@synthesize topSpeed = _topSpeed;
@synthesize nearestWormhole = _nearestWormhole;

- (void)setTopSpeed:(double)speed
{
    if ((speed < 1) && (speed > 0)) _topSpeed = speed;
}

- (void)orbitPlanet:(Planet *)aPlanet atAltitude:(double)km
{
    // put the code to orbit a planet here
    double speed = [self topSpeed];
    if (speed > MAX_RELATIVE) speed = MAX_RELATIVE;
    [[self nearestWormhole] travelToPlanet:aPlanet
                                   atSpeed:speed];
}

@end
```

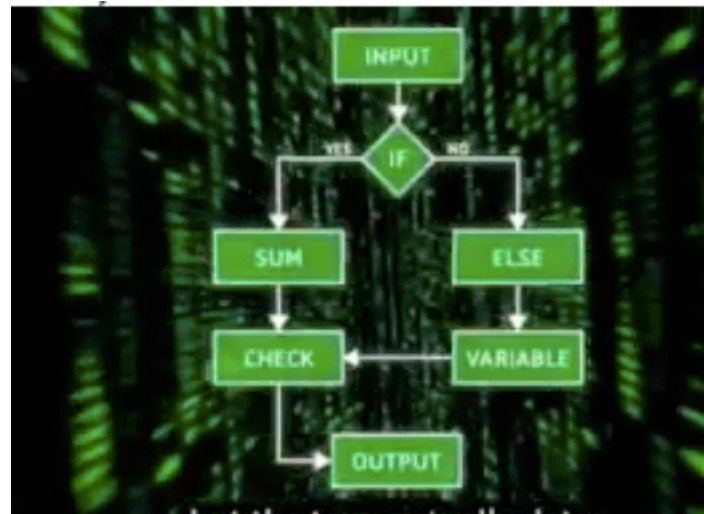
Square brackets inside square brackets.

Stanford CS193p  
Fall 2011

# Software Design

Computer  
Science

## Algorithms



# Algorithm Definition

algorithm | 'algə,riTHəm |

Procedure → Process

noun

a process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer: *a basic **algorithm** for division.*

## Algorithm

From Wikipedia, the free encyclopedia

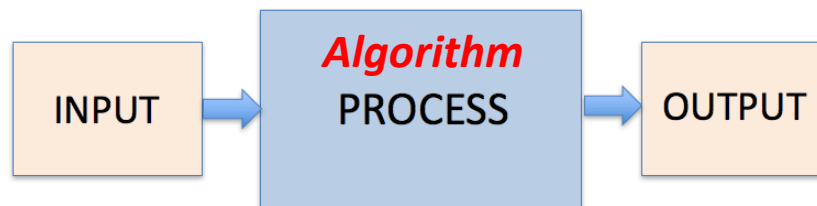
❖ Sequence of steps

- Deterministic
- NON-Deterministic

*For other uses, see [Algorithm \(disambiguation\)](#).*

In [mathematics](#) and [computer science](#), an **algorithm** (/ˈælɡərɪdəm/ ( listen) *AL-gə-ridh-əm*) is an unambiguous specification of how to solve a class of problems. Algorithms can perform [calculation](#), [data processing](#) and [automated reasoning](#) tasks.

An algorithm is an [effective method](#) that can be expressed within a finite amount of space and time<sup>[1]</sup> and in a well-defined formal language<sup>[2]</sup> for calculating a [function](#).<sup>[3]</sup> Starting from an initial state and initial input (perhaps [empty](#)),<sup>[4]</sup> the instructions describe a [computation](#) that, when [executed](#), proceeds through a finite<sup>[5]</sup> number of well-defined successive states, eventually producing "output"<sup>[6]</sup> and terminating at a final ending state. The transition from one state to the next is not necessarily deterministic; some algorithms, known as [randomized algorithms](#), incorporate random input.<sup>[7]</sup>



# Algorithm Definition

---

## ❖ Instructions (recipe)

- Step-by-step (control sequence)
- Solving a problem or performing a task

## ❖ Execution

- Executed in any *control sequence (processes)*
- Dictated by *control constructs (procedures)*

## ❖ Result

- Output (e.g., computations)
- Transformation (e.g., games, simulations)

## ❖ Properties

- Well-formed
- Optimal (in time/code space)
- Terminates (in finite time)
  - Stops (“Halts”)
  - Waits (for user input)



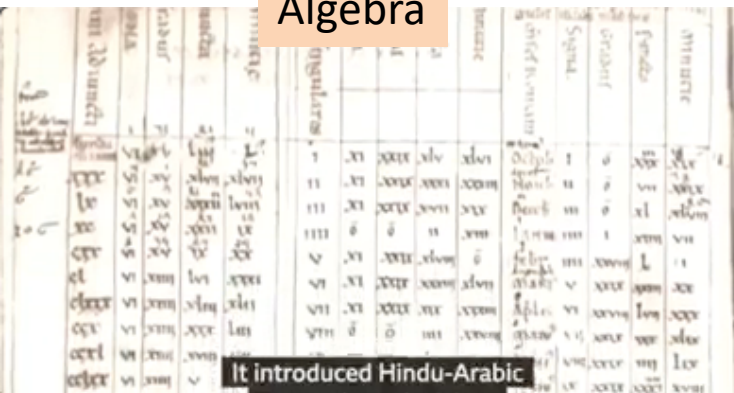
# History of Algorithms



The Hindu-Arabic number system,



Algebra

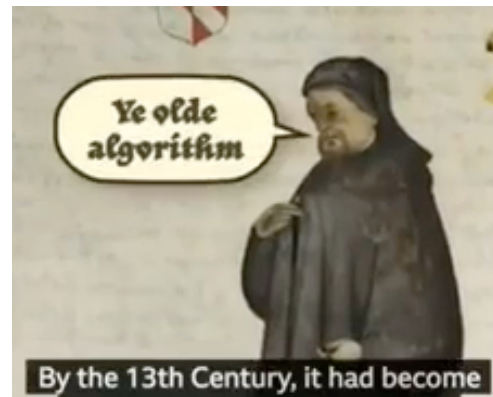


Alan Turing



al-Khwarizmi → algoritmi

- ❖ 9<sup>th</sup> century Persian mathematician
- ❖ 13<sup>th</sup> century usage by Chaucer
- ❖ 19<sup>th</sup> century intro'd
- ❖ 20<sup>th</sup> century first use in computing by Alan Turing



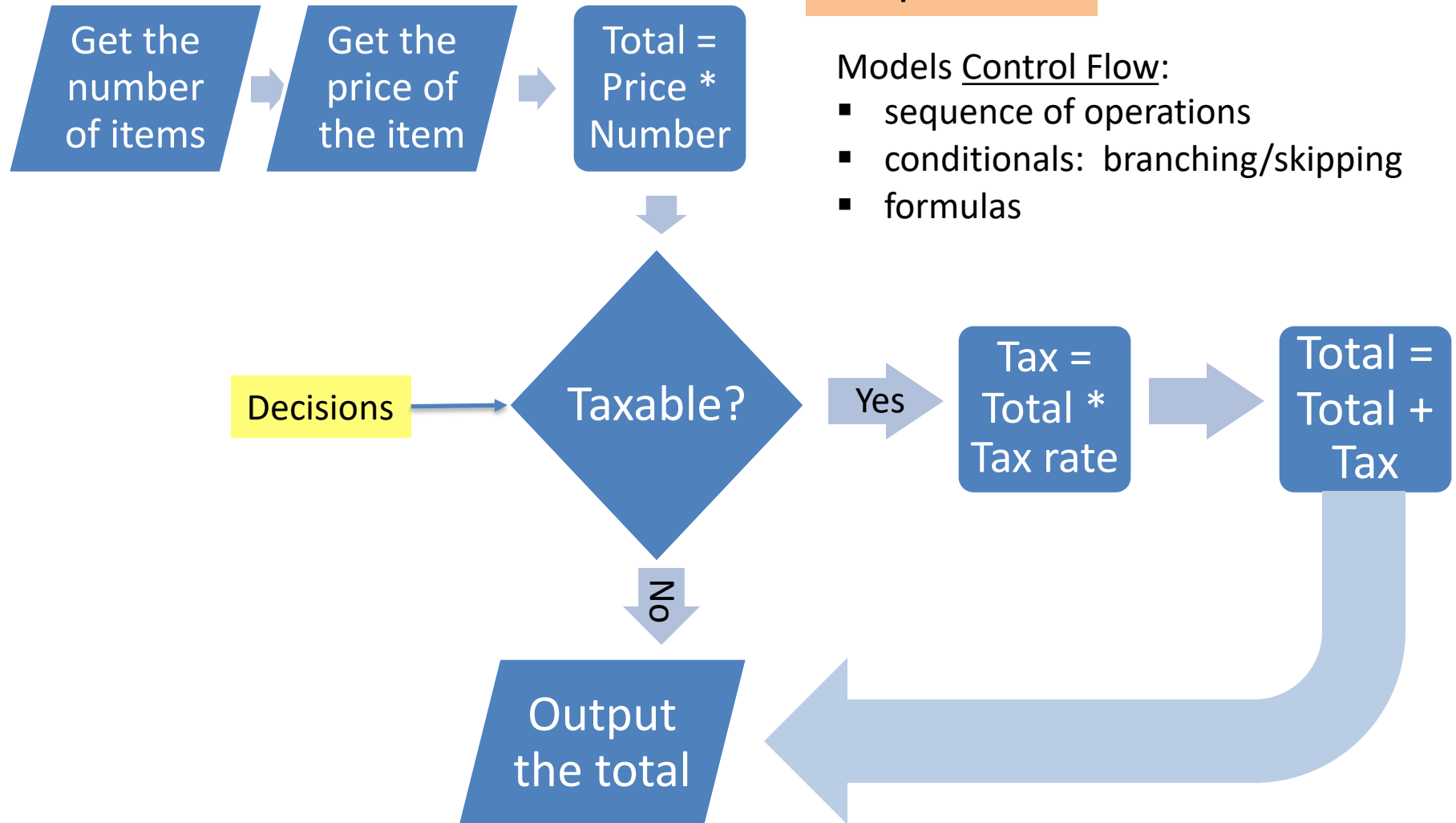


# Algorithms as Flowcharts

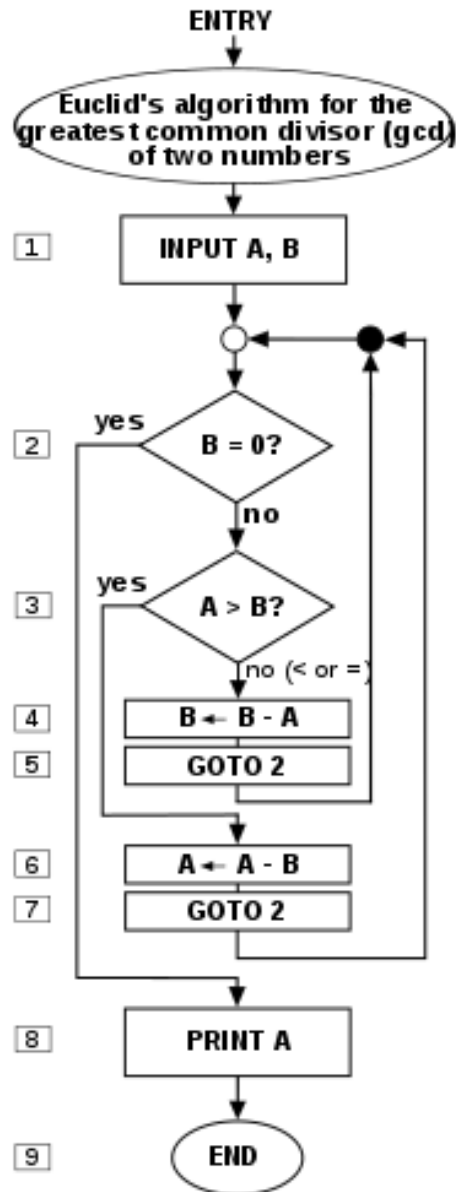
## Simple Sales

Models Control Flow:

- sequence of operations
- conditionals: branching/skipping
- formulas



# Algorithms as Flowcharts



- ❖ *Conditional* execution (decisions)
- ❖ *Iterative* execution (Loops)

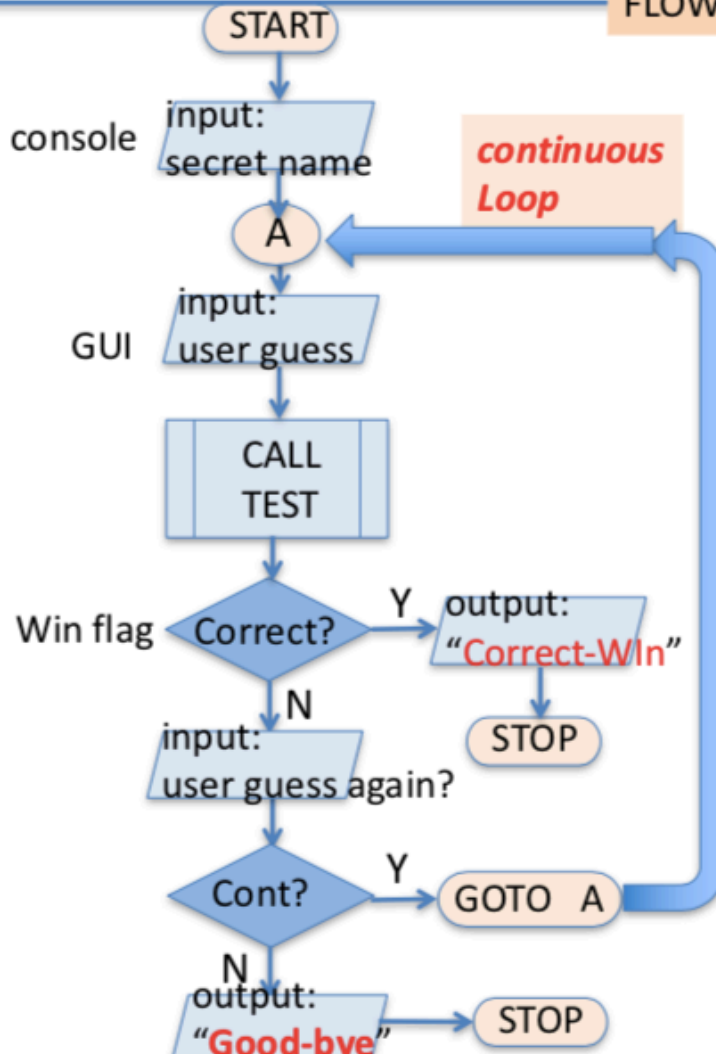
Flow chart of an algorithm (Euclid's algorithm) for calculating the greatest common divisor (g.c.d.) of two numbers  $a$  and  $b$  in locations named A and B. The algorithm proceeds by successive subtractions in two loops: IF the test  $B \geq A$  yields "yes" (or true) (more accurately the number  $b$  in location B is greater than or equal to the number  $a$  in location A) THEN, the algorithm specifies  $B \leftarrow B - A$  (meaning the number  $b - a$  replaces the old  $b$ ). Similarly, IF  $A > B$ , THEN  $A \leftarrow A - B$ . The process terminates when (the contents of) B is 0, yielding the g.c.d. in A. (Algorithm derived from Scott 2009:13; symbols and drawing style from Tausworthe 1977).

# Algorithms as Flowcharts

## Lab 2: Guess Secret Name

FLOW CHART

Part 2



Boolean Function



- ❖ Conditional execution (decisions)
- ❖ Iterative execution (Loops)

# Algorithm Structure

---

## ❖ Simple

- ☐ Single formula
- ☐ Select a formula from a set

## ❖ Complex

- ☐ Conditional set (IF-THEN-ELSE)
- ☐ Iterative set (loop)
- ☐ Nested set
- ☐ Subroutines/methods

➤ “Programs” embed “Algorithms” (0 to N)

# Most Important Algorithms

## ❖ Searching

### ☐ Binary

- List must be sorted

- We will **search** a homonym database that you create in Lab 5
- We will **sort** characters for anagrams in Lab 4

## ❖ Sorting

### ☐ Bubble

### ☐ Iterative-Binary Search

### ☐ Quick Sort

### ☐ Q-Sort

### ☐ Others

- 2/3 Sort (see next slide)

# Algorithms in Labs

---

1. Hello World: Input/Output
2. Secret word game: compare (guess to secret)
3. Temp convert: formula\*
4. Words/pals+anagrams: check, sort algorithms\*
5. Words/homonyms: Input database, check, search\*
6. Primes A: textbook algorithm\*
7. Primes B: Dr Jeff algorithm\* (compare results)
8. Tic-Tac-Toe: check win, next move algorithms\*

\*use “methods” (subroutines)

# Algorithms– Math

## ❖ Formulas

- Pythagorean Theorem
- Taylor Series
- Dr Jeff

## ❖ Algorithms – Historical

- Euclidean GCD
- Sieve of Eratosthenes
  - Prime numbers
  - Not most efficient
- Newton's Iteration
- Horner's Rule
- Calendars
- Pascal's Triangle
  - Polynomial coefficients
- Fibonacci Sequence
  - Natural model
  - Financial modeling

### **A Few Famous Algorithms from History**

Euclidean Algorithm (described in Euclid's Elements circa 300BC, see Wikipedia article) For finding the GCD of two integers

Sieve of Eratosthenes (276 BC - 195 BC)

For creating a table of prime numbers up to some chosen maximum

Newton's Iteration (Sir Isaac Newton, 1643-1727)

For estimating square roots

Horner's Rule (William George Horner, 1786-1837)

For quickly evaluating polynomials, especially polynomials of large degree



# Algorithms– Computing

## ❖ Donald Knuth

- Stanford Computer Science Prof.
- Books – 3 volumes
  - 1) Fundamental algorithms
  - 2) **Searching & Sorting** algorithms

COMP182

## Searching & Sorting

### ❖ Sorting

- ☐ Linear
- ☐ Bubble sort
- ☐ Binary sort
- ☐ Quick sort
- ☐ Qsort

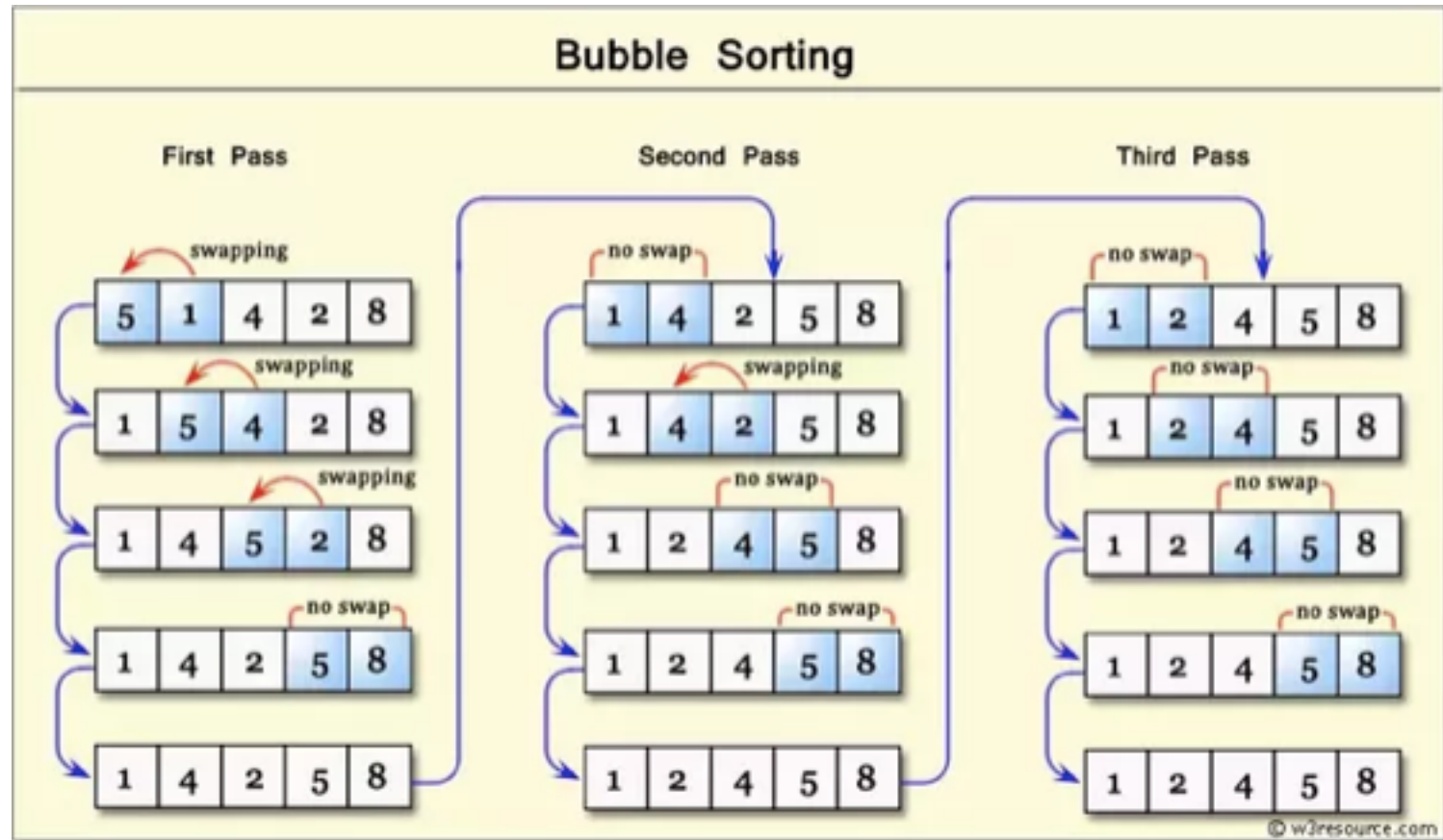
### ❖ Searching (sorted list)

- ☐ Binary search
- ☐ Indexed search
- ☐ Hashed search

### ❖ Cryptography

- ☐ Encryption
  - DES
  - AES
  - 3DES
- ☐ Hashing
  - Hash functions
  - MD5
  - SHA1

# Algorithms– Bubble Sort



# Algorithms– Prime Numbers

---

## Sieve of Eratosthenes

- Pick a value  $n$ .
- Write out a table of the integers from 2 to  $n$ .
- Cross out all entries that are multiples of 2.
- Find the smallest remaining number  $> 2$ , which is 3.
- Cross out all entries that are multiples of 3.
- Continue until you reach the floor of the square root of  $n$ .
- The numbers that remain are prime.

# Algorithms– Prime Numbers

Old **FORTRAN**

FIGURE 9-5 LIST OF PRIME NUMBERS

```
DO 5 I = 5,10000.2
  K = SQRT(REAL(I))
  DO 4 J = 3,K,2
    IF(MOD(I,J).EQ.0)GO TO 5
  4 CONTINUE
  WRITE(6,31)I
5 CONTINUE
STOP
31 FORMAT(1X,I6)
END
```

Skip all even numbers.

Determine  $\sqrt{I}$ . Argument of SQRT must be real.

Check if I is divisible by any integer  
up to and possibly including  $\sqrt{I}$ .

Number is prime; print it.

Test routine → *subroutine*

*nested*

“DO loops”

# Dr Jeff Algorithm

Lab 6

Dr Jeff *Optimized* algorithm

- ❖ Numbers = 1 .. 1000 → *odd* only
  - ❖ divisors = {P} = all **found primes** ≤ limit
  - ❖ limit = **sqrt** (Number)
  - ❖ can easily cast out 5s (...5, ...0) ➤ so only need to **start at 7**
- ← conjecture to be proven

# Algorithms– Prime Numbers

INTRO

Output: 1, 2, 3

Init: NUM = 3

*Find all prime numbers from 1 to N*

Dr Jeff **Optimized** algorithm

A

CALL  
TEST

PRIME?

Output: NUM

Incr: NUM += 2

odd # only

DONE?

GOTO A

STOP

TEST

Clear P flag

5? TEST div by 5

RETURN

easy

3? TEST div by 3

RETURN

{P}=all found primes  
≤ sqrt(NUM)

TEST div {P}

RETURN

SET P flag

RETURN

NUM ≥ N?

## INTRO

### ❖ Tomohiko Sakamoto's Algorithm

**Algorithm**, to find the day of week from any given date.

❖ Leap years (1/4)

❖ Leap centuries (1/400)

```
1 int dow(int y, int m, int d) {
2     static int t[] = {0, 3, 2, 5, 0, 3, 5, 1, 4, 6, 2, 4};
3     y -= m < 3;
4     return (y + y/4 - y/100 + y/400 + t[m-1] + d) % 7;
5 }
```

### ❖ Zeiler's Algorithm

**Zeller's algorithm** [\[edit\]](#)

Main article: [Zeller's congruence](#)

In Zeller's algorithm, the months are numbered from 3 for March to 14 for February. The year is assumed to begin in March; this means, for example, that January 1995 is to be treated as month 13 of 1994.<sup>[9]</sup> The formula for the Gregorian calendar is

$$w = \left( d + \left\lfloor \frac{13(m+1)}{5} \right\rfloor + y + \left\lfloor \frac{y}{4} \right\rfloor + \left\lfloor \frac{c}{4} \right\rfloor - 2c \right) \bmod 7,$$

### ❖ Basic Algorithm (0=Sat, 1=Sun, ...)

**Basic method for mental calculation** [\[edit\]](#)

This method is valid for both the [Gregorian calendar](#) and the [Julian calendar](#). Britain and its colonies started using the Gregorian calendar on Thursday, September 14, 1752; the previous day was Wednesday, September 2, 1752 ([old style](#)). The areas now forming the United States adopted the calendar at different times depending on the colonial power: Spain and France had been using it since 1582, while Russia was still using the [Julian calendar](#) when Alaska was purchased from it in 1867.

The formula is  $\left( d + m + y + \left\lfloor \frac{y}{4} \right\rfloor + c \right) \bmod 7$ , where: c=year in century=year mod 100, m is from table



# Complexity: 2/3 Sort

## What algorithms have the most unexpected **big-O time complexity**?



Michal Forišek, teacher, scientist, competitive programmer

Updated May 11, 2014 · Upvoted by Kiran Kannar, Master's Computer Science, University of California, San Diego (2018) and Alon Amit, CS degree and many years of coding.

There is a sorting algorithm, the **two-thirds sort** (a.k.a. **Stooge sort**), that works as follows:

If there are at most two elements, sort them directly using at most one swap.  
Otherwise:

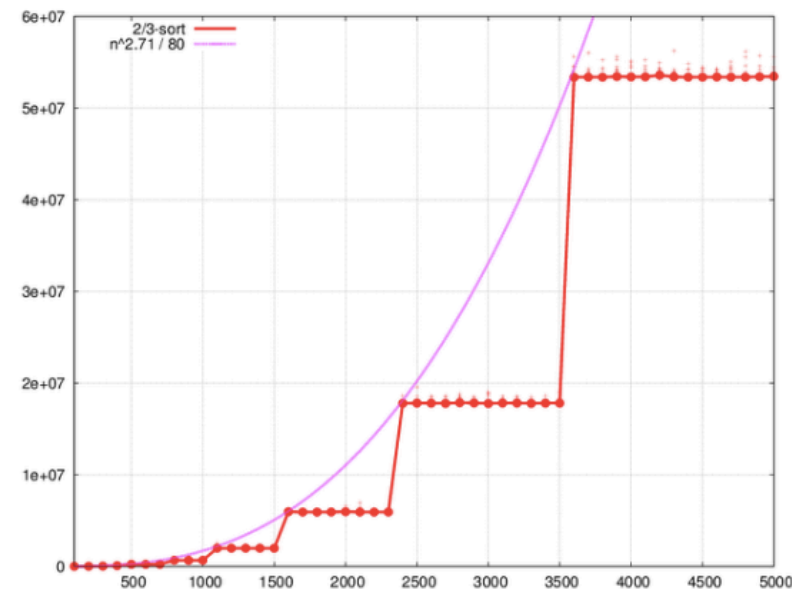
1. sort the first 2/3 of the array recursively
2. sort the last 2/3 of the array recursively
3. sort the first 2/3 of the array recursively

I like to use this algorithm in my lectures about time complexity, for several reasons:

First of all, **it's not immediately obvious that this algorithm actually sorts.**

Can you prove that? (Hint posted as a comment.)

But that is still not the thing that makes it the most interesting. The most interesting thing is its **actual time complexity**. You are probably used to algorithms where the time complexity is a smoothly growing function: the larger the input, the longer the running time. Well, here's how the running time of this particular algorithm looks like:



The plot shows the actual running time (red) and the asymptotic estimate (magenta) as a function of input. While the asymptotic upper bound is a nice smooth function, the actual running time is **surprising**: it looks like a staircase.

# Top 10 Algorithms

- **Plagiarism detection, using Rabin Karp String matching**
  - String matching algorithms are pervasive in software. One particularly fun one, is Rabin Karp, which is used in Plagiarism detection. As a student in CS (or in any major), plagiarism detection should be of interest ;-)
  - Rabin Karp is relatively easy to implement. See this: [Rabin-Karp algorithm - Wikipedia](#) ↗
  - Rabin Karp has also inspired a string matching routine in Zlib (one of the most popular un/zip libraries ever). See [this](#) ↗, directly into the source code.
- **Matching users to servers, using Gayle-Shapely Algorithm for Stable Marriage problem**
  - This is a beautiful algorithm for fair matching. Simple, elegant and effective. In its core form, it's also straightforward to implement. Has numerous applications. See: [Stable marriage problem - Wikipedia](#) ↗
- **A toy implementation of Viterbi algorithm**
  - Ubiquitous in cell phone technology, and many other applications, Viterbi algorithm is a Dynamic Programming based algorithm that finds the most *likely* sequence of states.
  - See this toy implementation: <http://homepages.ulb.ac.be/~dgon...> ↗
- **Music Search using Fast Fourier Transforms (FFT)**
  - Music recognition is done by converting it into frequency domain using FFT. FFT has implementations in number of languages. See this article for a great start: [Shazam It! Music Recognition Algorithms, Fingerprinting, and Processing](#) ↗.

# Top 10 Algorithms

- Implement **RSA algorithm**
  - SSL transport, is the bane of safe existence on Internet these days. One of the most well-known algorithms in secure transport, is RSA, named by the first initials of its inventors.
  - Implementing RSA is fun and instructive e.g. [C code to implement RSA Algorithm\(Encryption and Decryption\)](#) ↗
- **Safe Browsing (or similar) using Bloom filters**
  - Bloom filters found very rare usage until the world got more online and we hit scale. But these days, we see new applications very frequently.
  - Chrome browser uses Bloom filters to make preliminary decision on [safe browsing](#) ↗. See some novel applications [here](#).
- Implement an **LALR parser**
  - As a CS student, you may have already implemented it as part of your compiler's class. But if not, then you should. LALR parsing makes syntactic sense of source code, whichever language you use
  - Many implementations of LALR exist. e.g. [Where can I find a \\_simple\\_, easy to understand implementation of an LR\(1\) parser generator?](#) ↗
  - Also, use YACC to understand LALR parsing better.
- **Treemap using Red Black Trees!**
  - RB Trees are not algorithms, but they are famed enough, that no discussion of tantalizing DS/Algorithms is complete without discussing them.
  - The smoothest way to see/implement RB Trees, is to look at [Treemap](#) ↗ implementation in Java.

# Top 10 Algorithms

---

- **Circle Drawing using Bresenham's algorithm**
  - Ever wondered, how circles are drawn on the screen, with minimal jaggedness (aliasing)? Bresenham's elegant algorithm is at play here. See a version here: [Circle Generation Algorithm](#) ↗ .
  - A refreshing use of a similar algorithm, is to make properly sized **tabs in Chrome**. ↗ Something we see almost every day. Such hidden gems!
- **Implement PageRank**
  - Can't miss this. This transformed our lives in ways we never thought possible. Get started here: [Pagerank Explained Correctly with Examples](#) ↗



# Algorithms

$q(i | j, k) = u(i, j, k) / \text{SUM}[u(n, j, k)]$

$i=\text{race}, j=\text{surname}, k=\text{census block}$

## BRILLIANT MATH OR JUNK SCIENCE?

Federal regulators are using Marc Elliott's algorithm to crack down on discriminatory lending. The GOP doesn't like it.

BY JAMES RUFUS KOREN

Marc Elliott didn't know he'd become a player in the financial world until he received an unexpected email from a friend.

It read simply, "Did you know you just cost Ally Financial \$80 million?"

Until that moment nearly three years ago, the Rand Corp. statistician hadn't known an algorithm he'd devised years earlier for healthcare research had found its way from Rand's headquarters in Santa Monica to the halls of a powerful financial regulator in Washington, D.C.

Or that the agency, the Consumer Financial Protection Bureau, had used his breakthrough formula to underpin racial discrimination allegations against auto lending companies, starting with former General Motors lending arm Ally Financial, which paid \$80 million to settle in

### Big numbers

The Consumer Financial Protection Bureau has applied Elliott's algorithm to reach settlements with several big auto lenders:

**\$21.9 million**

Toyota, 2016

**\$24 million**

Honda, 2015

**\$80 million**

Ally Financial, 2013

2013.

"My first reaction was just that it had really moved along," said Elliott, 49, who has spent much of his nearly 21 years at Rand researching healthcare issues, not finance. "I hadn't been aware at all."

And it's gone much further since then.

If you have a credit card, a car loan or almost any type of debt other than a mortgage, there's a chance your name and address have been run through Elliott's algorithm, a complex formula that crunches data from the Census Bureau.

But as it has become more widely used, Elliott's work and the CFPB's application of it have found their way into the middle of a fight between the federal consumer watchdog and politicians who want to scrap the agency. Some congressional Republicans have gone so far as to call the CFPB's use of Elliott's system "junk science."

[See Elliott, C7]

# Rubik's Cube Algorithm

L is left side

R is right side ... [See More](#)

**HOW TO SOLVE THE RUBIK'S CUBE**

L	R2	B	L'	R	U2	F2	L	B
D	U2	B	L'	R2	B2	L'	D2	U
L'	B2	U2	R2	B	D2			

**A UNIVERSAL SOLUTION**

▶ ● ——— -3:49 ⚙️ 📺 🔍 🔊

NOW I'VE SEEN EVERYTHING

# My Math Theorem

$$\forall m,n: |mn - nm| = |m-n| * 9$$

where m, n are decimal digits (0..9)

Examples:

$$|m-n|=1$$

$$21 - 12 = 9$$

$$32 - 23 = 9$$

$$43 - 34 = 9$$

$$54 - 45 = 9$$

Examples:

$$|m-n|=2$$

$$31 - 13 = 18$$

$$42 - 24 = 18$$

$$53 - 35 = 18$$

$$64 - 46 = 18$$

Examples:

$$|m-n|=3$$

$$41 - 14 = 27$$

$$52 - 25 = 27$$

$$63 - 36 = 27$$

$$74 - 47 = 27$$

Examples:

$$|m-n|=4$$

$$51 - 15 = 36$$

$$62 - 26 = 36$$

$$73 - 37 = 36$$

$$84 - 48 = 36$$



# My Other Math Theorem

square of any integer that ends in **5**:  
form of  $n5^2 = n(n+1)$  **25**

$$\forall n: ((n*10) + 5)^2 = n(n+1)*100 + 25$$

where n is any decimal integer

Examples:

$$n = 2$$

$$25^2 = 625$$

$$n = 3$$

$$35^2 = 1225$$

$$n = 4$$

$$45^2 = 2025$$

$$n = 5$$

$$55^2 = 3025$$

# Algorithms

Computer  
Science

## Cryptography

➤ See separate slide set

# Cryptography

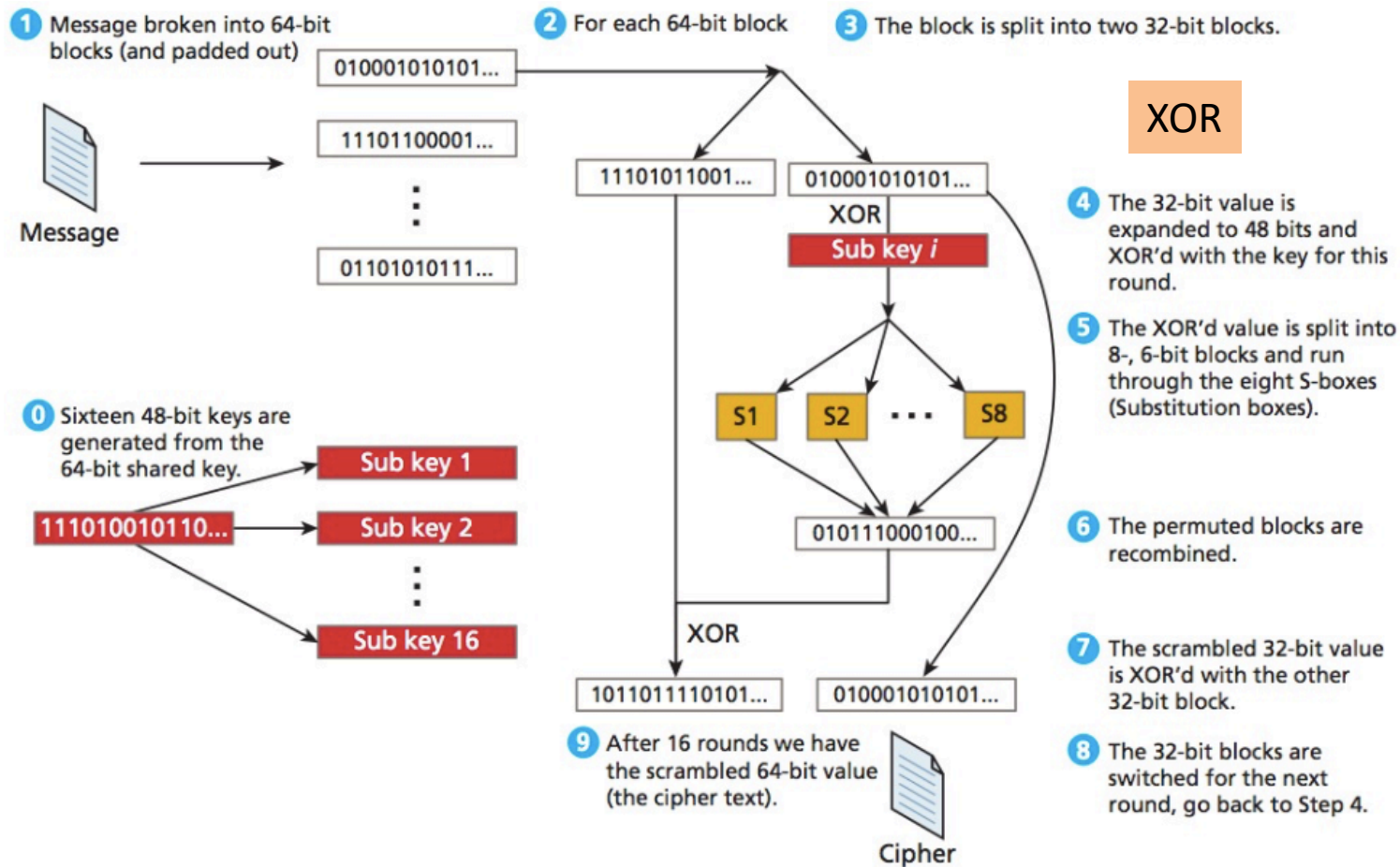
## ❖ Encryption

- ☐ Used to secure data in storage & *transit*
- ☐ Many standards (DES, 3DES, etc.)
- ☐ algorithms use sequence of XOR operations
- ☐ use public-private key pairs
- ☐ replaces each character in situ with a code
- ☐ data retains same length
- ☐ does not detect tampering

## ❖ Hashing

- ☐ Used to secure data in storage (only)
- ☐ A few standards (MD, SHA)
- ☐ algorithms use complex sequence of math operations with key
- ☐ use private keys *derived* from random issued words
- ☐ does not replace data
- ☐ adds a “hash” value to each block of data
- ☐ hash value is a fixed 160 bits for SHA
- ☐ detects tampering (*raison d’etre*)

# DES Encryption

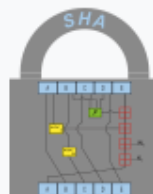


**FIGURE 16.10** High-level illustration of the DES cipher

Algorithms + Keys

# SHA Hashing

## Secure Hash Algorithm



### Concepts

hash functions • SHA • DSA

### Main standards

SHA-0 • SHA-1 • SHA-2 • SHA-3

## SHA-1

From Wikipedia, the free encyclopedia

In [cryptography](#), **SHA-1** (**Secure Hash Algorithm 1**) is a [cryptographic hash function](#) which takes an input and produces a 160-bit (20-byte) hash value known as a [message digest](#) - typically rendered as a [hexadecimal](#) number, 40 digits long. It was designed by the United States [National Security Agency](#), and is a U.S. [Federal Information Processing Standard](#).<sup>[3]</sup>

Since 2005 SHA-1 has not been considered secure against well-funded opponents,<sup>[4]</sup> and since 2010 many organizations have recommended its replacement by [SHA-2](#) or [SHA-3](#).<sup>[5][6][7]</sup> Microsoft, Google, Apple and Mozilla have all announced that their respective browsers will stop accepting SHA-1 [SSL certificates](#) by 2017.<sup>[8][9][10][11][12][13]</sup>

In 2017 [CWI Amsterdam](#) and [Google](#) announced they had performed a [collision attack](#) against SHA-1, publishing two dissimilar PDF files which produced the same SHA-1 hash.<sup>[14][15][16]</sup>

# SHA Hashing

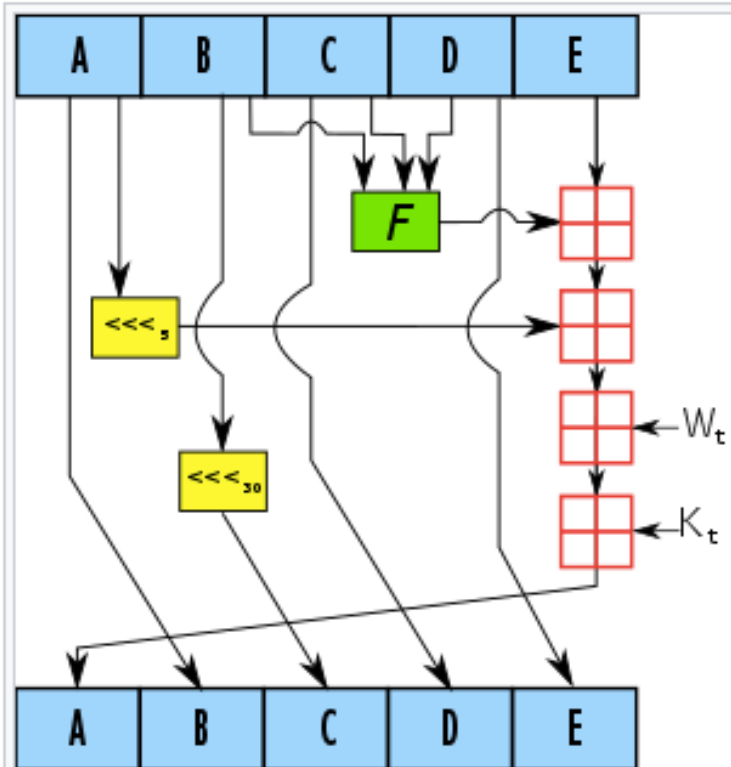
## SHA-1

## General

<b>Designers</b>	National Security Agency
<b>First published</b>	1993 (SHA-0), 1995 (SHA-1)
<b>Series</b>	(SHA-0), SHA-1, SHA-2, SHA-3
<b>Certification</b>	FIPS PUB 180-4, CRYPTREC (Monitored)

### Cipher detail

<b>Digest sizes</b>	160 bits
<b>Block sizes</b>	512 bits
<b>Structure</b>	Merkle–Damgård construction
<b>Rounds</b>	80



One iteration within the SHA-1 compression function:

A, B, C, D and E are 32-bit words of the state:

$F$  is a nonlinear function that varies:

$\lll_n$  denotes a left bit rotation by  $n$  places;

$n$  varies for each operation;

$W_t$  is the expanded message word of round  $t$ :

$K_t$  is the round constant of round  $t$ ;

$\boxplus$  denotes addition modulo  $2^{32}$ .

# SHA Hashing

## INTRO

Comparison of SHA functions

Algorithm and variant		Output size (bits)	Internal state size (bits)	Block size (bits)	Max message size (bits)	Rounds	Operations	Security bits (Info)	Capacity against length extension attacks	Performance on Skylake (median cpb) <sup>[57]</sup>		First Published
										long messages	8 bytes	
MD5 (as reference)		128	128 (4 × 32)	512	Unlimited <sup>[58]</sup>	64	And, Xor, Rot, Add (mod 2 <sup>32</sup> ), Or	<64 (collisions found)	0	4.99	55.00	1992
SHA-0		160	160 (5 × 32)	512	2 <sup>64</sup> − 1	80	And, Xor, Rot, Add (mod 2 <sup>32</sup> ), Or	<34 (collisions found)	0	≈ SHA-1	≈ SHA-1	1993
SHA-1								<63 (collisions found <sup>[59]</sup> )		3.47	52.00	1995
SHA-2	SHA-224	224	256 (8 × 32)	512	2 <sup>64</sup> − 1	64	And, Xor, Rot, Add (mod 2 <sup>32</sup> ), Or, Shr	112	32 0	7.62	84.50	2004 2001
	SHA-256	256						128		7.63	85.25	
		SHA-384	384	512 (8 × 64)	1024	2 <sup>128</sup> − 1	80	And, Xor, Rot, Add (mod 2 <sup>64</sup> ), Or, Shr	192	128 (≤ 384)	5.12	
	SHA-512	512	256						0	5.06	135.50	
	SHA-512/224	224						112	288	≈ SHA-384	≈ SHA-384	
	SHA-512/256	256						128	256			
SHA-3	SHA3-224	224	1600 (5 × 5 × 64)	1152	Unlimited <sup>[60]</sup>	24 <sup>[61]</sup>	And, Xor, Rot, Not	112	448	8.12	154.25	2015
	SHA3-256	256		1088				128	512	8.59	155.50	
	SHA3-384	384		832				192	768	11.06	164.00	
	SHA3-512	512		576				256	1024	15.88	164.00	
		SHAKE128	d (arbitrary)	1344		min(d/2, 128)	256	7.08	155.25			
	SHAKE256	d (arbitrary)	1088		min(d/2, 256)	512	8.59	155.50				



# Software

---



Computer  
Science

# Theory

## Halting problem

From Wikipedia, the free encyclopedia

In [computability theory](#), the **halting problem** is the problem of determining, from a description of an arbitrary [computer program](#) and an input, whether the program will finish running or continue to run forever.

[Alan Turing](#) proved in 1936 that a general [algorithm](#) to solve the halting problem for *all* possible program-input pairs cannot exist. A key part of the proof was a mathematical definition of a computer and program, which became known as a [Turing machine](#); the halting problem is *undecidable* over Turing machines. It is one of the first examples of a [decision problem](#).

Informally, for any program *f* that might determine if programs halt, a "pathological" program *g* called with an input can pass its own source and its input to *f* and then specifically do the opposite of what *f* predicts *g* will do. No *f* can exist that handles this case.

Unfortunately, Turing proved that such a program can't exist, which means we can't just fire all the math professors. His proof is a proof by contradiction: he assumes you *could* create such a program, and shows that this leads to a result that is "absurd"/"obviously false".

In particular, he says, if you wrote such a program, you could write a new program that uses it:

```
1 my program:
2   if ("my program" halts)
3     run forever
4   otherwise
5     halt
```

Java

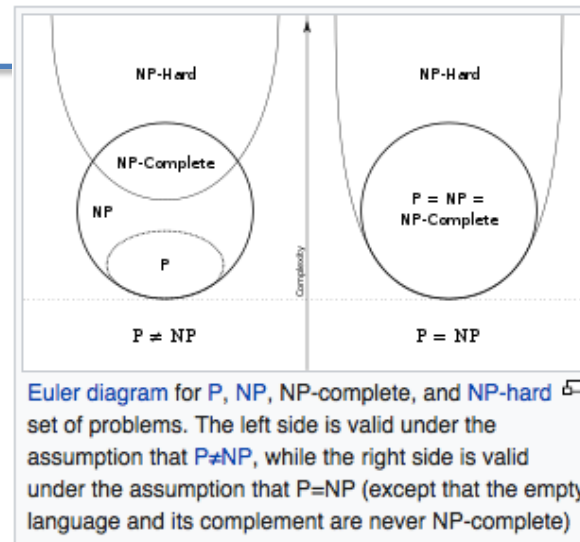
```
while(true) {
    System.exit(0);
}
```

In other words, 'my program' will run forever if that halting program says that it halts, and it will halt if the halting program says it will run forever — no matter what the halting program says, it has to be wrong!

## NP-completeness

From Wikipedia, the free encyclopedia  
(Redirected from [NP complete](#))

Not Provable?



In computational complexity theory, an **NP-complete** decision problem is one belonging to both the **NP** and the **NP-hard** complexity classes. In this context, **NP** stands for "**nondeterministic polynomial time**". The set of NP-complete problems is often denoted by **NP-C** or **NPC**.

Although any given solution to an NP-complete problem can be verified quickly (in polynomial time), there is no known efficient way to locate a solution in the first place; indeed, the most notable characteristic of NP-complete problems is that no fast solution to them is known. That is, the time required to solve the problem using any currently known **algorithm** increases very quickly as the size of the problem grows. As a consequence, determining whether it is possible to solve these problems quickly, called the **P versus NP problem**, is one of the principal **unsolved problems in computer science** today.

While a method for computing the solutions to NP-complete problems using a reasonable amount of time remains undiscovered, **computer scientists** and **programmers** still frequently encounter NP-complete problems. NP-complete problems are often addressed by using **heuristic** methods and **approximation algorithms**.

# Software

---

## Code Structure

# Documentation

---

## Documentation – COMMENTS

- ❑ Maintainability
- ❑ Code (conditional use)

### ❖ IN line

foo xxx foo 'comments (VB)  
foo //comments (JS, others)

### ❖ Beginning of line

'comment line (VB)  
//comment line (JS, others)

### ❖ Multi line

/\* comments line 1  
comments line 2  
comments line 3 \*/

# Comments in Java

---

```
// This is an example of a single line comment using two slashes

/* This is an example of a multiple line comment using the slash and asterisk.
   This type of comment can be used to hold a lot of information or deactivate
   code, but it is very important to remember to close the comment. */

package fibsandlies;
import java.util.HashMap;

/**
 * This is an example of a Javadoc comment; Javadoc can compile documentation
 * from this text. Javadoc comments must immediately precede the class, method, or field being documented.
 */
```

# Software = Code + Data

---

- ❖ All applications projects integrate *both* CODE and DATA
- ❖ CODE manages the DATA
- ❖ CODE is *combined* into a single file (.exe, .class)
- ❖ DATA can be I/O or a *stored collection*, in any format:
  - I/O (console, GUI) – not stored
  - Set of simple files:
    - ✧ CSV as .txt or other (.drj)
  - Formal DBMS (SQL)



# Software

## ❖ Control

- ❑ Control Flow
  - Confined to structures
- ❑ Control structures
  - Subroutines/Methods
  - IF-THEN-ELSE
  - LOOPS
    - FOR (iteration)
    - WHILE
    - DO-WHILE

## ❖ Data

- ❑ Data Flow
  - Input
  - Output
- ❑ Data structures
  - Files
  - **Arrays**
  - Structures
  - Databases

# Code

# Data



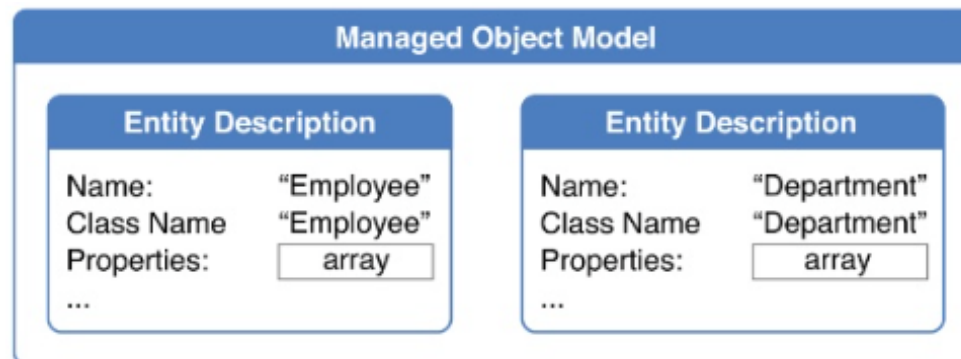
# Software Structure-*Data*

▼	and.. ▼	Street (Home Address) ▼	City (Home...
↕		12733 Parkyns St	Los Angeles
↕		2586 Northlake Cir	Westlake Vlg
↕		21629 Wo He Lo Trail	Chatsworth
↕		5560 Oak Park Ln #307	Oak Park
↕		14930 Magnolia Blvd #7	Sherman Oaks
↕	& Elie Daher	1562 N Courtney Ave	Hollywood
↕		1628 N Courtney Ave	Hollywood
↕	& Claude	104 S Young Rd	Payson
↕		7826 Winnetka Ave	Canoga Park
↕	& Michael		Hills
↕	& Daniel		
↕			

Data

## DATA STRUCTURE

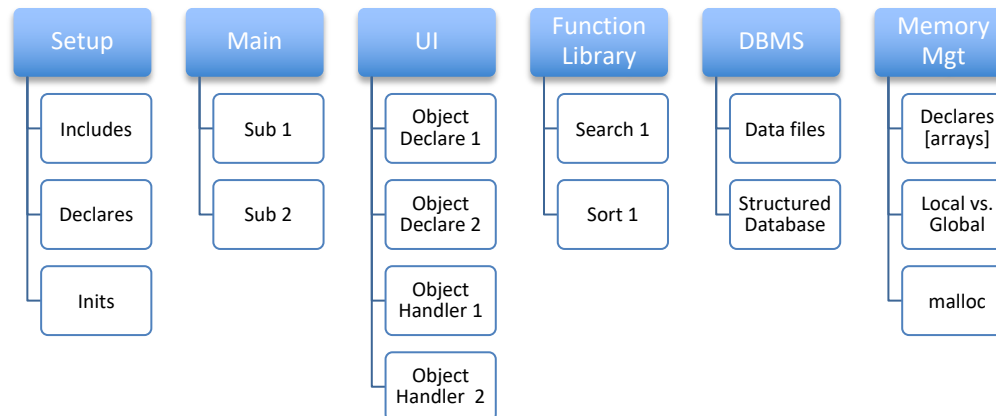
- ☐ Simple data files
- ☐ Data models
  - ◆ Apple Core Data
- ☐ RDBMS



# Software Structure-*Code*

```
s {  
    database that retrieves all entities name "Entity."  
    t *request = [NSFetchRequest fetchRequestWithEntityName  
ed object context.  
venience for alloc] init].  
ctContext *managedObjectContext = [NSManagedObjectContext  
tent store coordinator to the managed object context.  
tContext setPersistentStoreCoordinator:self.persistentS  
s an error object.  
r;  
  
and fetch the results.  
cts = [managedObjectContext executeFetchRequest:request  
with the res  
ect count: s
```

## Code



# Code Structure-*Sections*

## BOOK

- ☐ Forward
- ☐ Preface
- ☐ Introduction
- ☐ **Chapters**
  - 
  - 
  -
- ☐ Conclusion
- ☐ Index

*Read sequentially*

## PROGRAM

- ☐ Includes
- ☐ Declarations
- ☐ Initialization
- ☐ **Blocks**
  - *Classes*
  - *Procedures*
    - *Functions*
    - *Event Handlers*
    - *File Handlers*
    - *I/O Handlers*
    - *Error Handlers*
- ☐ Documentation

*Executed NON-sequentially  
(by thread of control -- "behavior")*

# Code Structure-C

```
/* C program */
#include <stdio.h> //standard C I/O library

void main ( )
{
    // inits
    Int a,b

    /* C program
    Goes here */

    printf ("Hello"\n) //print to std output (console)
    scanf ("%d%d", &a,&b) //input from keyboard

}
```

# Code Structure-*Java*

---

```
/* Java program */
import javax.swing.*; //standard Java library
// "system" does not need to be imported

public class helloWorld {
    public static void main (String[] args) {
        system.out.println ("Hello world!");
    }
}
```



# Other Languages

---

- C
- VB

# Code Structure-*Main (VB)*

Imports System.IO 'to Read/Write Files  
Imports System.Drawing.Printing 'to use a Printer

SETUP

Public Class Form1

Inherits System.Windows.Forms.Form 'to use a "Window"

< Declarations >

Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

< Initialization code >

End Sub

< MODEL code>

< CONTROLLER code>

End Class

MAIN

# Code Structure-*Subs (VB)*

```
Private Sub limra(ByRef dd As Int16, ByVal x As Byte)
    < statements >
End Sub
```

```
Private Sub abs(ByRef dd As Int16)
    < statements >
End Sub
```

```
Private Function pad0(ByVal dd)
    < statements >
    Return (dd)
End Function
```

```
Private Sub caller()
    Call abs(x)
    Call pad0(n)
    Call limra(z,m)
End Sub
```

## ❖ Call

## ❖ Parameter passing

- By *Value*
- By *Reference*

## ❖ Return

- Sub -> *void*
- Function -> *value*

# Subroutines in VB

```
Private Sub limra(ByRef dd As Int16, ByVal x As Byte)
    < statements >
End Sub
```

```
Private Sub abs(ByRef dd As Int16)
    < statements >
End Sub
```

```
Private Function pad0(ByVal dd)
    < statements >
    Return (dd)
End Function
```

```
Private Sub caller()
    Call abs(x)
    Call pad0(n)
    Call limra(z,m)
End Sub
```

❖ Call

❖ Parameter passing

- By Value
- By Reference

❖ Return

- Sub -> void
- Function -> value

# Subroutines in C

```
void limra(dd, ff) {  
    int a,b;  
    < statements >  
}
```

SUBROUTINE

```
int limfn(dd, ff) {  
    int a,b;  
    < statements >  
    Return (a);  
}
```

FUNCTION

```
void main() {  
    int a,b,c;  
    limra(a,b);  
    x = limfn(b,c);  
}
```

MAIN PROGRAM  
Calls subs

- ❖ Template
- ❖ Call
- ❖ Parameter passing
  - By *Value*
  - By *Reference*
- ❖ Return
  - Sub -> *void*
  - Function -> *value*

# Methods in Java

```
public static void limra(dd, ff) {  
    int a,b;  
    < statements >  
}
```

SUBROUTINE

```
public static int limfn(dd, ff) {  
    int a,b;  
    < statements >  
    Return (a);  
}
```

FUNCTION

```
public static void main(String[] args) {  
    int a,b,c;  
    limra(a,b);  
    x = limfn(b,c);  
}
```

MAIN PROGRAM  
Calls subs

- ❖ Template
- ❖ Call
- ❖ Parameter passing
  - By *Value*
  - By *Reference*
- ❖ Return
  - Sub -> *void*
  - Function -> *value*

# Compiler Directives in C

## ❖ INCLUDE

```
#include <stdio.h>  
#include <p18f4321.h>
```

## ❖ MACRO (compare to EQU vs. subroutines)

```
#define portc0 PORTCbits.RC0
```

## ❖ PRAGMA (compare to ORG)

- ❖ `#pragma code` begin
- ❖ `#pragma code` int\_vect = 0x00000008



# Actual Code-*Visual Basic*

## INTRO

VB

## SETUP

**Imports** System.IO

**Imports** System.Drawing.Printing

Public **Class** Form1

**Inherits** System.Windows.Forms.Form

**\*\*\*system constants**

Public Version As String = "Version x.x"

Dim DataVer As String 'ver # in file

Dim copyr As String = "Copyright(c) 2009-14"

**\*\*\*system switches**

Dim DEMO As Boolean = False, REL As Boolean = False 'EDIT for Release/debug

Dim DEBUG As Boolean = False

**\*\*\*screen X,Y positions**

Dim cprtX As Int16 = 488, cprtY As Int16 = 444, demoX As Int16 = 344, demoY As Int16 = 0

Dim verX As Int16 = 177, verY As Int16 = 0, DverX As Int16 = 177, DverY As Int16 = 16

**\*\*\*standard vars**

Dim i, j, k, l, m, n As Byte

Dim u, v, w, x, y, z As Int16

Dim ss, tt, uu, vv, ww, xx, yy, zz As Single

Dim wstr, xstr, ystr, zstr, xxstr, alertst, srchstr, matchstr As String

Dim vobj, wobj, xobj, yobj, zobj As Object

Dim err1 As Boolean = False, err2 As Boolean = False

**\*\*\*file**

Dim OpenFileDialog As OpenFileDialog

Dim streamxx As StreamReader

Dim file\_root As String = "C:\Users\Jeff\Documents\Jeff's files\DrJeff Software\Word World\

# Actual Code-*Visual Basic*

VB

## Init – LOAD DATA

*\*\*\*LOAD Files*

*'\*Main Load data*

Private Sub loadF(ByVal fnam As String)

Try

FileOpen(1, fnam, OpenMode.Input)

Catch ex As Exception

Call filerr("error opening data file in <loadF> - " & ex.Message)

End Try

End Sub

*'\*Load User data*

Private Sub loadU(ByVal fnam As String)

Call loadhead(fnam, "User")

If cc Then Call loadfav() Else favlen = 0 : Call filerr("Favorites not loaded")

Call loadchex() : ccx = cc : If Not cc Then Call filerr("Checks not loaded")

Call loadpat() : If Not cc Then plen = 0 : Call filerr("Channel patches not loaded")

FileClose(1)

If Not ccx Then 'use default chex

For i = 0 To catmax

k = genlen(i) : k = limra0(k, gmax)

For j = 0 To k

chex(i, j) = 1 : Next : Next

End If

End Sub

# Actual Code-*Visual Basic*

VB

Private Sub SavBtn\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles SavBtn.Click

cc = True

If favlen < 1 Then Call actok("Saving Empty Favorites")

If cc Then Call savall(LoadfilexU)

End Sub

\*\*\*Box handlers

Private Sub chg\_box(ByVal ar() As String, ByVal ln As Int16, ByRef obj As System.Object)

Dim ii As Int16

obj.Items.Clear()

If ln <= 0 Then Return

'Call dialert("len=" & CStr(ln))

Try

For ii = 0 To ln - 1

xstr = ar(ii) '& "\*\*\*"

'Call dialert(xstr)

If xstr = "" Then xstr = "<blank>"

If obj Is ListBox3 Then

Call wordwrap(ListBox3, xstr, 21)

Else

obj.Items.Add(xstr)

End If

Next

Catch ex As Exception

Call Palert("error in <chg\_box>" & " -" & ex.Message)

End Try

End Sub

## UI – User Events

# Actual Code-*Visual Basic*

VB

## FUNCTION LIBRARY

**\*\*\*SEARCH/SORT**

**\*\*\*search**

Private **Function** searchlist(ByRef listx As Object, ByVal xx As String, ByVal ln As Byte)

*'\*find index of xx in sorted list*

Private **Function** search1D(ByVal ar(), ByVal ln, ByVal xs)

Private **Function** srchcond(ByRef ss) *'process search text; set cond2*

**End Function**

**\*\*\*sort**

Private **Sub** sortadd(ByRef nam() As String, ByRef box As Object, ByVal ln As Byte)

*'\*bubble Dn last in sorted list, nam(0:ln); gen xar()*

Private **Sub** sortlist(ByRef listx As Object, ByVal ar() As String, ByVal ln As Byte)

*'\*gen index array 'xar' fm sorted list*

Private **Sub** sortarr(ByRef ar() As String, ByVal ln As Byte)

*'\*sort array via 'xar' –str*

**End Sub**

# Code Guidelines

---

## ❖ Scope

- **Local** – best to use – **Private**
- **Global** – *be very careful* – **Public**

## ❖ Type casting

- Use *explicit* types (avoid implicit casting & *overloading*)

## ❖ Procedure parameter passing

- Use “By Value” for variables
- Use “By Reference” for objects

## ❖ Condition codes

- Set “CC” binary var (T/F) on action completion
- Test “CC” before continuing with next action

## ❖ Error trapping & handling

- **TRY** & **CATCH** blocks – use generously
- Catch exception descriptions
- Add as much pertinent info as possible (esp. location)
- Report via “alert boxes”
- Never allow un-trapped errors – they cause program interruption  
(that is what “beta testing” is for)

# Tradeoffs

## ❖ Memory

### ☐ Code (KB-MB)

- Static
- Lines of code
- Verbosity

VS

### ☐ Data (MB-GB-TB)

- Small files (CSV)
- Databases (SQL)
- Big data (data mining)

## ❖ Performance (Speed)

### ☐ Total execution time (sec)

- Small tasks (compute only)
- Big simulations (e.g., weather)
- Verbosity

### ☐ User response (msec)

- Clicks
- Text characters
- Forms

### ☐ Embedded control (msec)

- Real-time response
- Interrupts

# Object Oriented Design

Ch 9

➤ major properties of OOP

## ❖ Encapsulation

- ❑ Objects
  - ✧ Classes as models
- ❑ Classes
  - ✧ Properties
  - ✧ **Constructors**
  - ✧ Methods

```
Class Foo
  <decl vars (init)>
  Fn 1
  Fn 2
  <code>
End Class
```

- ❖ Declare class Foo
- ❖ Declare vars
- ❖ Define *methods*
- ❖ Add code

## ❖ Inheritance

- ❑ Class Instantiations

```
Foo Fee
  Fn 3
  <other code>
End Class
```

- ❖ Fee **Instantiates** Foo
- ❖ Fee **Inherits** Foo
- ❖ Fee **Adds code to** Foo  
*polymorphism*

## ❖ Polymorphism

- ❑ Multiple Instantiations
  - ✧ Small changes to Methods or code

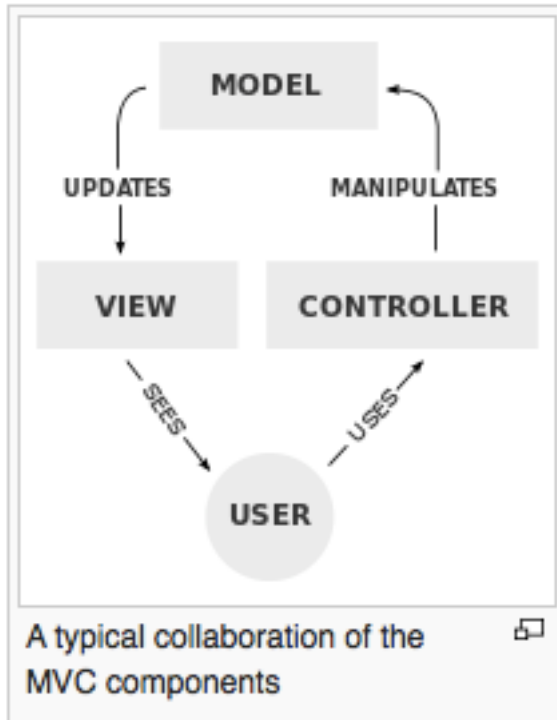


# Software

---

## Design Patterns

# Design Patterns: MVC



## ❖ Microsoft

- Visual Studio
  - ❖ Design View
  - ❖ Code Behind

## ❖ Apple

- Xcode
  - ❖ Storyboard
  - ❖ Code

- View Controller** - A controller that supports the fundamental view-management model in iOS.
- Navigation Controller** - A controller that manages navigation through a hierarchy of views.
- Table View Controller** - A controller that manages a table view.
- Tab Bar Controller** - A controller that manages a set of view controllers that represent tab bar items.
- Split View Controller** - A composite view controller that manages left and right view controll...
- Page View Controller** - Presents a sequence of view controllers as pages.



# MVC

## ❖ Model

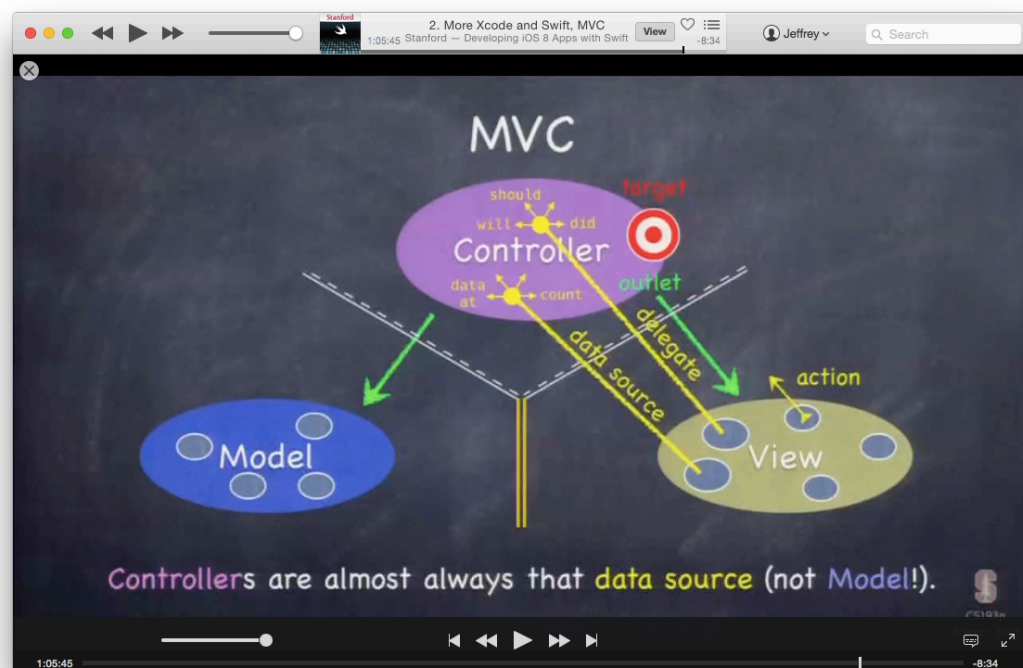
- ❑ Application function set

## ❖ View

- ❑ Design – what user sees

## ❖ Controller

- ❑ Event handlers (user actions)



# Software

---

## Web Applications

# Web Apps vs. Web Design

## Web Apps

### ❖ WHAT

- ☐ **Functions**
- ☐ **Features**
- ☐ **Benefits**

### ❖ Implement

- ☐ *Code* Structure
- ☐ Programming Languages
- ☐ Coding (programming)

## Web UI

### ❖ HOW IT LOOKS

- ☐ Appearance
- ☐ Features
- ☐ Benefits

### ❖ Design

- ☐ *Site* Structure
- ☐ Page layout
- ☐ Styles
- ☐ Visual elements
  - ✧ Photos
  - ✧ Videos
  - ✧ Audio/sound



# Client-Server Model



# Client Devices

## ❖ Phones

RESPONSIVE/ADAPTIVE LAYOUTS

- ☐ iPhone (Apple iOS)
- ☐ Android (Samsung, HTC, Motorola → Google)
- ☐ Windows (Nokia, Microsoft?)
- ☐ Blackberry (ex-RIM, dying)

RUN BROWSERS

## ❖ Tablets

- ☐ iPad (Apple iOS)
- ☐ Android (Samsung, Google *Chromebook*)
- ☐ Amazon (Kindle, Fire)

## ❖ Hybrids (optional keyboard, stylus)

- ☐ Microsoft *Surface*
- ☐ Apple iPad *Pro* (iOS)

## ❖ PCs

- ☐ Windows PCs (HP, Dell, Toshiba, et al.)
- ☐ Mac (Apple Mac Pro, MacBook)

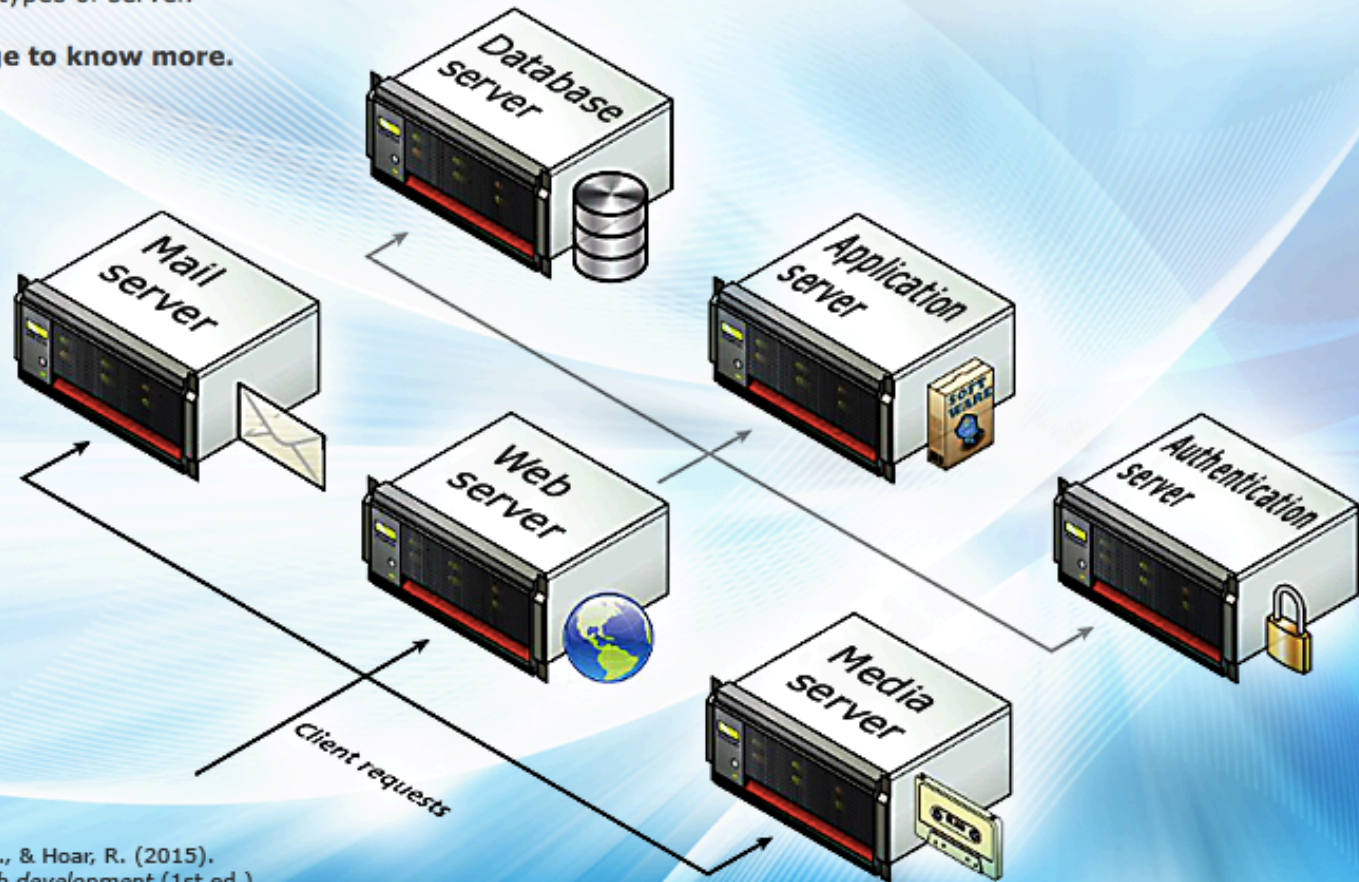


# Server Types

## Client-Server Model

Most real-world Web sites are served by many servers. It is common to split the functionality of a Web site between several different types of server.

Click each image to know more.



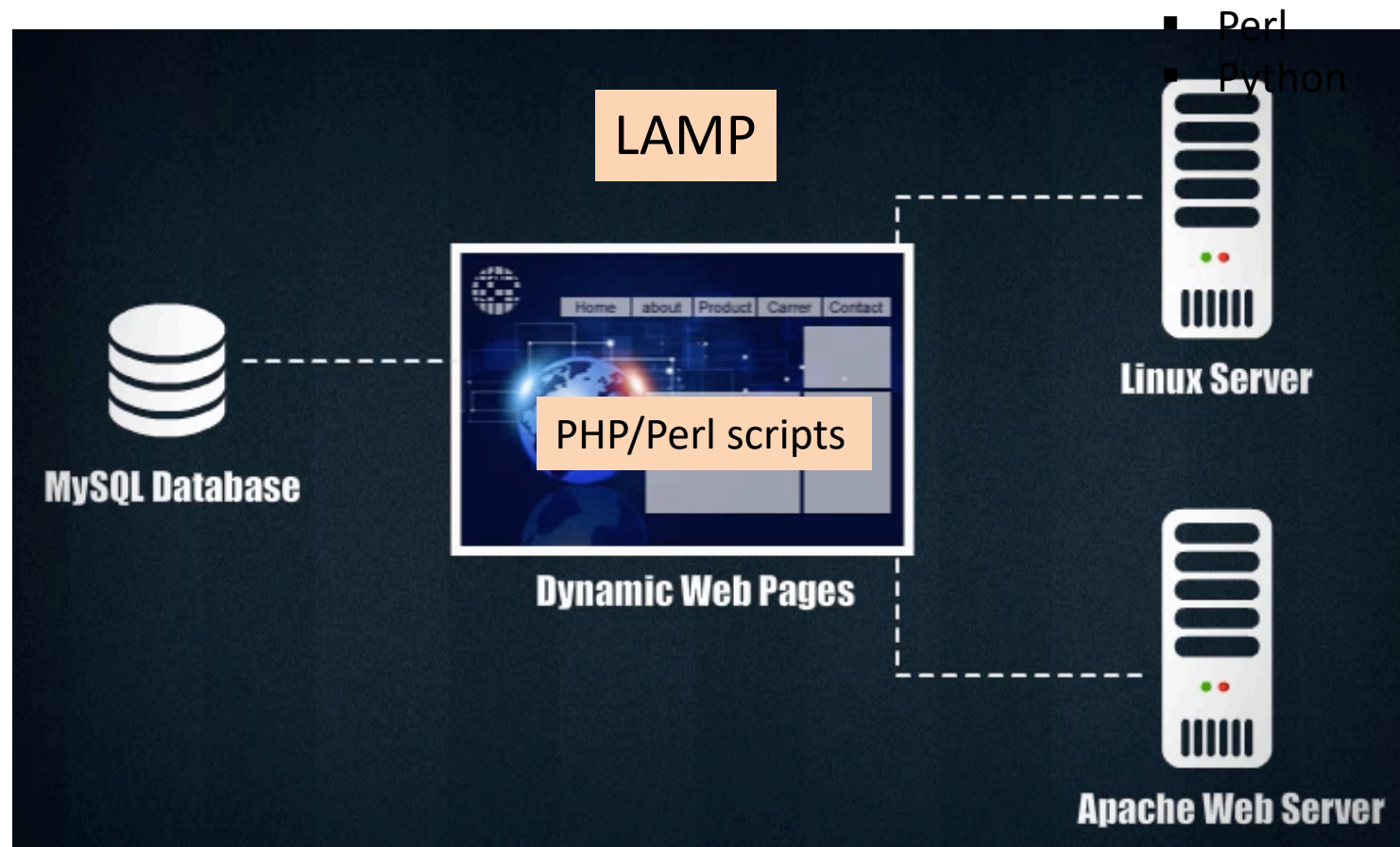
**Source:** Connolly, R., & Hoar, R. (2015).  
*Fundamentals of web development* (1st ed.).  
Upper Saddle River, NJ: Pearson.



# LAMP Overview

## Design of Web Apps

**LAMP = Linux + Apache + MySQL + PHP**



# Web Apps

## ❖ OS

- ☐ Linux (Ubuntu, Fedora)
- ☐ Windows

## ❖ Application server

- ☐ **Apache**
- ☐ **Windows IIS**

LAMP = Linux + Apache + MySQL + PHP

## ❖ Websites

- ☐ Pages (markup)
  - HTML
  - CSS
- ☐ **Applications** (functional)
  - PHP
  - Perl
  - Python
  - Java
  - Javascript (J2EE, JSON)
  - Ruby
  - ASP.NET

## ❖ Databases

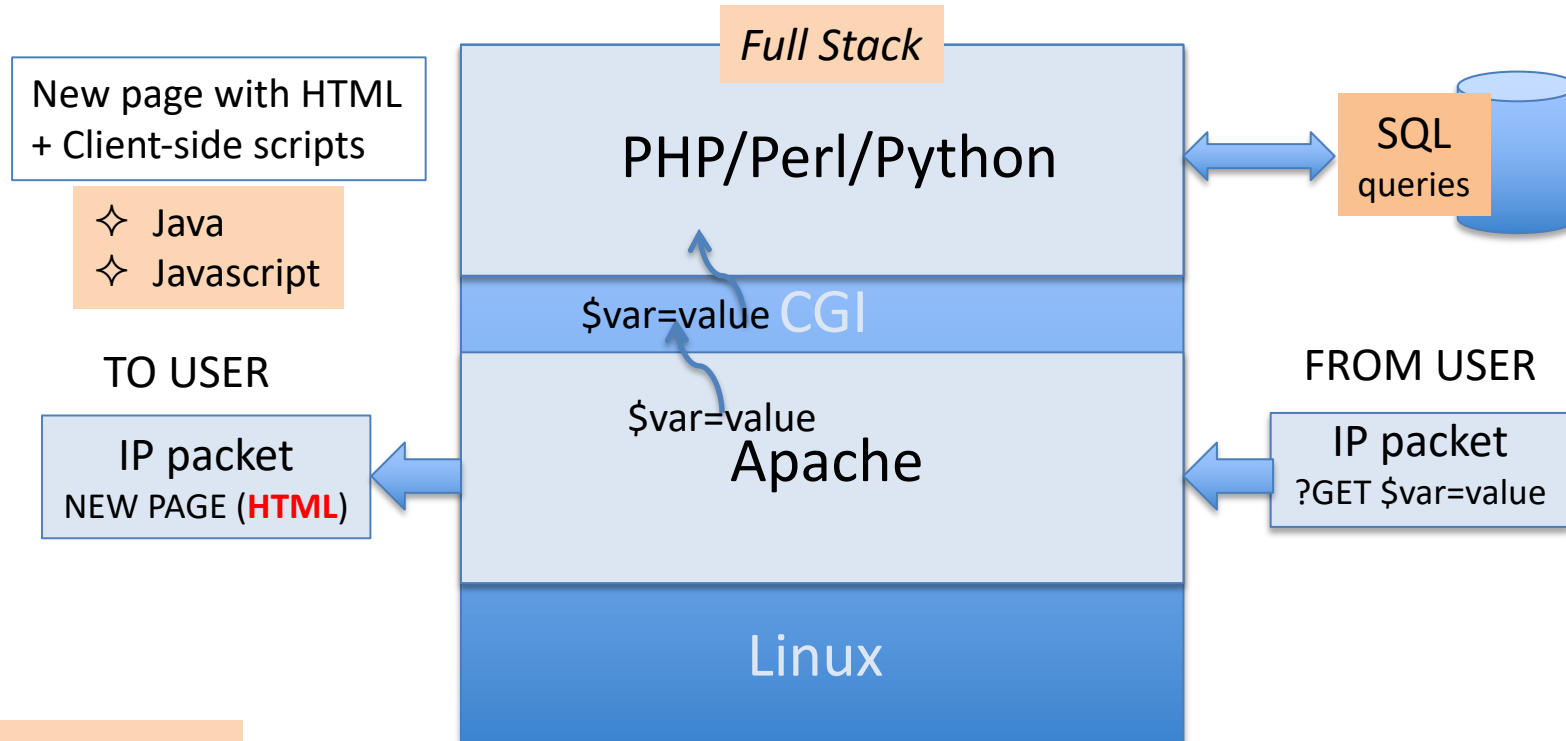
- ☐ MySQL
- ☐ MS SQL Server
- ☐ Others
  - Oracle
  - SAP
  - Teradata

# Web App Layers

Server-side

LAMP

**LAMP = Linux + Apache + MySQL + PHP**



## CGI in HTTP

[https://www.facebook.com/jeff.drobman/posts/10207546329763616?notif\\_t=like](https://www.facebook.com/jeff.drobman/posts/10207546329763616?notif_t=like)

[https://www.facebook.com/seadebido/posts/10208511988945552?notif\\_t=close\\_friend\\_activity](https://www.facebook.com/seadebido/posts/10208511988945552?notif_t=close_friend_activity)

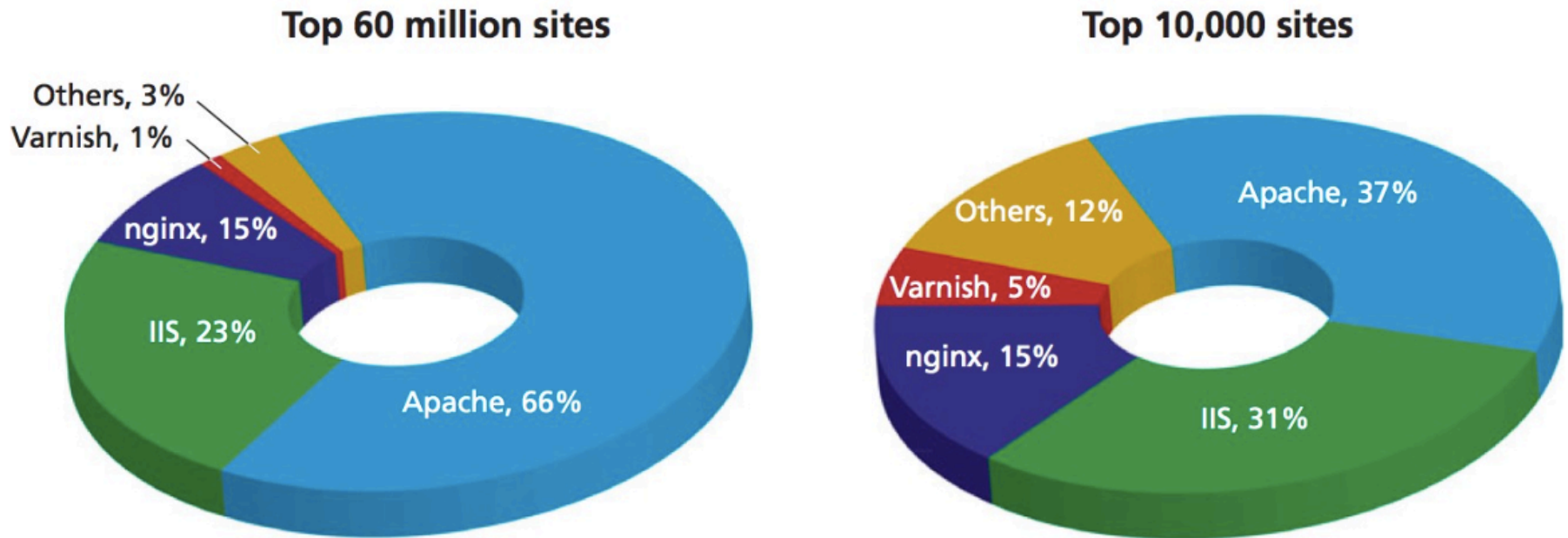
<https://portal.itt-tech.edu/shared/library/Pages/SchoolOfIT.aspx>

[https://en.wikipedia.org/wiki/Common\\_Gateway\\_Interface](https://en.wikipedia.org/wiki/Common_Gateway_Interface)

[http://www.amazon.com/gp/goldbox?ie=UTF8&ref\\_=br\\_isw\\_strs-2](http://www.amazon.com/gp/goldbox?ie=UTF8&ref_=br_isw_strs-2)

[https://secure.bankofamerica.com/myaccounts/signin/signIn.go?returnSiteIndicator=GAIMW&langPref=en-us&request\\_locale=en-us&capturemode=N&newuser=false&bcIP=F](https://secure.bankofamerica.com/myaccounts/signin/signIn.go?returnSiteIndicator=GAIMW&langPref=en-us&request_locale=en-us&capturemode=N&newuser=false&bcIP=F)

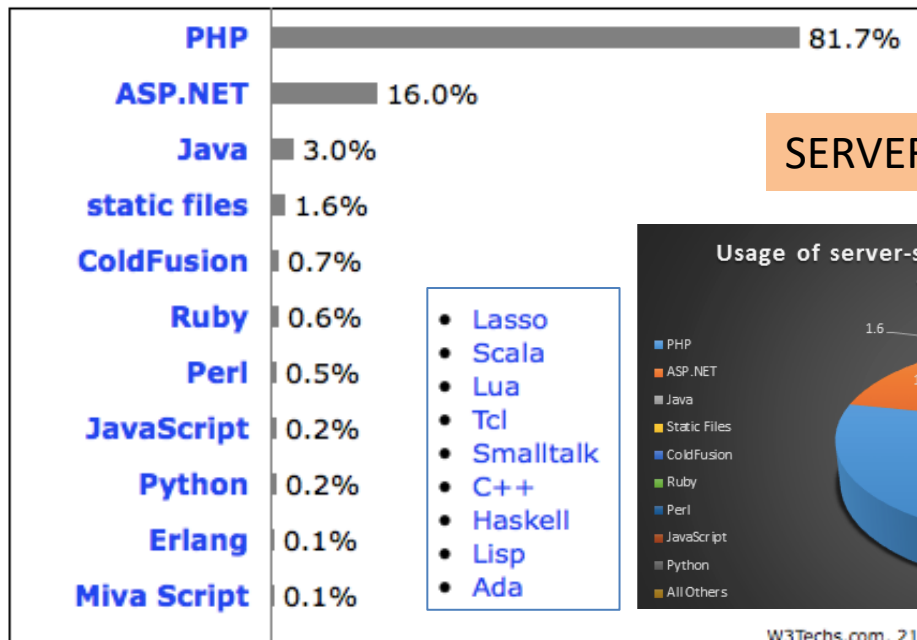
# Web Server Software



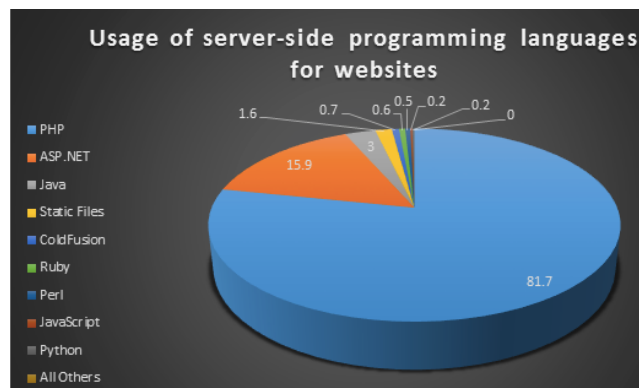
**FIGURE 19.11** Web server popularity (data courtesy of BuiltWith.com)

- ❖ Apache
- ❖ Microsoft IIS

# Web Languages

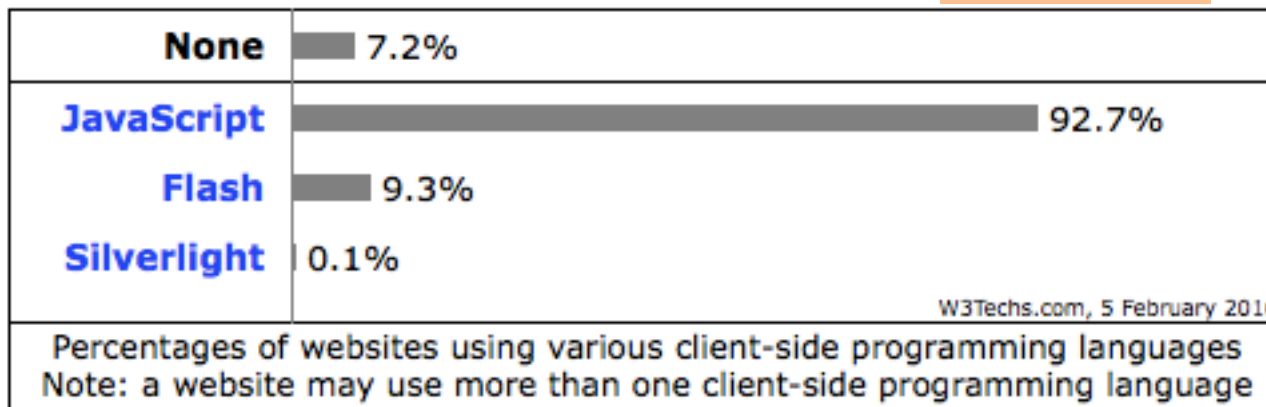


## SERVER Side



W3Techs.com, 21

## CLIENT Side



W3Techs.com, 5 February 2016

## Spoken Languages



```
<?php
$user = "Randy";
?>
<!DOCTYPE html>
<html>
<body>
<h1>Welcome <?php echo $user; ?></h1>
<p>
The server time is
<?php
echo "<strong>";
echo date("H:i:s");
echo "</strong>";
?>
</p>
</body>
</html>
```

**LISTING 8.1** PHP tags

## ❖ Hybrid web language

- ❑ Markup (tags)
- ❑ Application (OOP)

```
<!DOCTYPE html>
<html>
<body>
<h1>Welcome Randy</h1>
<p>
The server time is <strong>02:59:09</strong>
</p>
</body>
</html>
```

**LISTING 8.2** Listing 8.1 in the browser



# Web Language Variants

## LAMP Stack:

- PHP and MySQL
- OOP with PHP
- MVC with PHP (CodeIgniter)
- Cloud Server Management
- Basic JavaScript
- Ajax in PHP

## MEAN Stack:

Advanced JavaScript

MongoDB

Express

AngularJS

Node.js

Socket.IO

Redis

## Ruby on Rails:

- HAML, SASS, CoffeeScript
- Ajax in Rails
- Rails + Node.js
- Test Driven Development
- Heroku

For learning? Learn all the cool stuff.

1. Ruby's Rails, Sinatra, Cramp and Volt.
2. Python's Django and Flask.
3. Node's Express, Meteor and Sails.
4. PHP's Laravel, CakePHP, Yii and Symfony etc.
5. Java's Spring and Play.
6. Scala's Play and Lift.
7. Go's Beego.

All of these companies (and many more not listed, like Intel, HP and IBM) have published either extensive reports or blog articles on how they evaluated all the technology out there and switched to Node for either their flagship products or their new products...

Walmart

ebay

PayPal

DOW JONES

intuit

NETFLIX

LinkedIn

The New York Times

Microsoft

UBER

YAHOO!

Kingfisher

# Web File Types (MIME)

- ❖ .html (or .htm) – primary **markup** language
- ❖ .css – CSS **stylesheet** file
- ❖ .php – PHP **programming** language code
- ❖ .java – Java **interpreted programming** language code
- ❖ .js – Javascript **scripting** language code
- ❖ .pl – Perl **programming** language code
- ❖ .aspx – Microsoft .NET **programming** language (Active Server Pages)



## NOTE

**MIME** (multipurpose Internet mail extensions) types are identifiers first created for use with email attachments.<sup>10</sup> They consist of two parts, a type and a subtype, which together define what kind of file an attachment is. These identifiers are used throughout the web, and in file output, upload, and transmission.

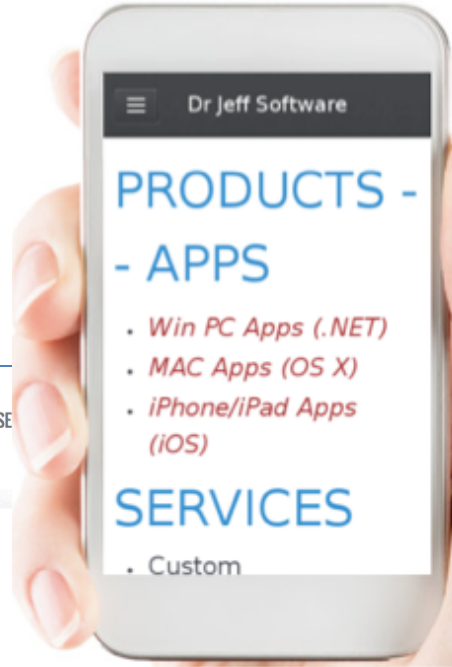


# Web Design for Mobile

## All about SCREEN SIZE and FORMAT

### ❖ ADAPTIVE

- ❑ Size
  - Phone
  - Tablet
- ❑ Orientation
  - Portrait
  - Landscape



### ❖ Apple Xcode

- ❑ Built-in coding for re-sizing
  - Phone
  - Tablet
- ❑ Orientation
  - Automatic
  - Simulator displays both

# Web Design Toolsets

## ❖ Wordpress

- ✧ Blog style
- ✧ **Templates**
- ✧ Plug-ins

WordPress

**NOT COVERED IN THIS COURSE**

To: Jeff Drobman

[Dr Jeff Software] Your site has updated to WordPress 4.4.1

Howdy! Your site at <http://www.pagerentals.me> has been updated automatically to WordPress 4.4.1.

You also have some plugins or themes with updates available. Update them now:  
<http://www.pagerentals.me/wp-admin/>

## ❖ Weebly

- ✧ Drag & drop

## ❖ Wix

Wix.com

Create Your Own Beautiful Website

Your stunning website is just a few clicks away. It's easy and free with Wix.



100s of Templates



Get Your Own Domain



Easy Drag n' Drop



Mobile Optimized

## ❖ Squarespace

- ✧ Pre-fab – **Templates**

web.com




Wix.com

weebly

network  
solutions

eHost.com

# Website Builders

30 Years of Experience			
	<b>Enhanced Build it For Me options</b> ★★★★☆ Rate it! (5552) ✓ Website built free - no upfront charges ✓ Templates: 100+	<b>9.5</b> Web.com Review	<a href="#">Visit Site »</a> \$1.95-\$3.95 »
	<b>Top-notch professional designs</b> ★★★★☆ Rate it! (2757) ✓ Completely FREE & customizable plan ✓ Templates: 500+	<b>9.5</b> Wix Review	<a href="#">Visit Site »</a> \$4.08-\$24.92 »
	<b>Intuitive drag and drop builder</b> ★★★★☆ Rate it! (1915) ✓ Anytime money-back guarantee ✓ Templates: 100+	<b>9.1</b> Weebly Review	<a href="#">Visit Site »</a> \$8.00-\$25.00 »

## TOP10BEST WEBSITE BUILDERS

Home > How to create a website > Business > Wix vs squarespace which is best for you

## Wix vs. Weebly vs. Squarespace: The Choice is Yours

December 23, 2015 / By Top10 Staff

# Web Hosts-Domains

❖ Yahoo

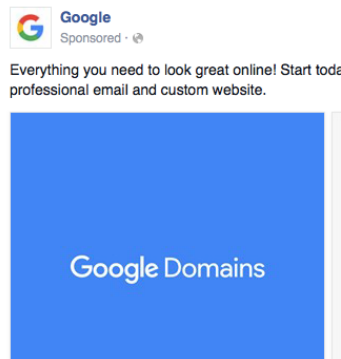
❖ Google

❖ GoDaddy

❖ iPage

❖ BlueHost

❖ SquareSpace



.com	49.7%
.ru	5.1%
.net	4.7%
.org	4.4%
.de	3.1%
.jp	2.1%
.uk	2.0%
.br	1.9%
.pl	1.5%
.in	1.3%
.it	1.3%
.fr	1.2%
.au	1.1%
.info	1.1%
.nl	1.0%
.cn	0.9%
.ir	0.8%
.es	0.7%
.cz	0.6%
.biz	0.6%
.kr	0.6%

TLDs



# Software

---

# Assembly Level

# Instruction Set Groups

## Computation

- ❖ ALU
  - ADD
  - SUB
  - AND
  - OR
  - XOR
  - NOT
- ❖ MULT [opt]
- ❖ DIV (rare)
- ❖ BIT
  - SET
  - CLR
  - TEST
- ❖ SHIFT
  - SHIFT
  - ROTATE

## Memory

- ❖ Reg-Mem
  - LOAD
  - STORE
  - MOV
- ❖ Mem-Mem
  - MOV
- ❖ Stack
  - PUSH
  - POP

## Program Control

- ❖ JUMP
  - JUMP/GOTO
- ❖ BRANCH
  - BRA
  - BRCC
- ❖ CALL
  - CALL/CALR
  - RET/RFI/RETFIE
- ❖ NOP

## I/O

- ❖ I/O
  - IN
  - OUT
- ❖ Mem Mapped
  - MOV PORT

## System Control

- ❖ Reset
  - RESET
- ❖ Power
  - SLEEP/HALT

# ALU & MOVE

PIC 18 MCU

## ADD

*Sets all flags*

**ADDLW** <data>;Add W to Literal <data> → **W** (only)

**ADDWF(C)** <addr>, **W/F** ;Add W to F (Data RAM) → **W or F**

## MOVE

*Sets NO flags (1 exception\*)*

**MOVLW** <data> ;Load Literal <data> → **W**

**MOVF\*** <addr>, **W/F** ;Load F at <addr> → **W or F** (same location; **sets N, Z**)

**MOVWF** <addr> ;Store **W** → **F** at <addr>

**MOVFF** <addr1>,<addr2> ;Move **F1** → **F2** in DataRAM (different location)

## MULTIPLY

*Sets NO flags*

**MULLW** ;Multiply W by Literal <data> → **PROD** [H,L]

**MULWF** ;Multiply W by **F** at <addr> → **PROD** [H,L]

**MOVFF** PRODL,<addr> ;Store PRODL\* → **F** at <addr>

**MOVFF** PRODH,<addr> ;Store PRODH\* → **F** at <addr>

\*SFR



# Addressing Modes

PIC 18 MCU

❖ Direct (in instruction) **MOVWF <addr>**[8]

**JUMP long-addr** [21], **BRA offset** [+7]

❖ Immediate (Literal Data) **MOVLW k** [-128 to +127]

❖ Indirect (Register indirect, uses FSR)

- **LFSR n,<addr>** [12-bit] (n=0, 1, 2)
- **MOVWF INDF<sub>n</sub>**
- **CLRF/MOVF POSTINC<sub>n</sub>/POSTDEC<sub>n</sub>/PREINC<sub>n</sub>**

❖ Indexed (Base Register FSR + Index Register W )

- **CLRF/MOVF PLUSW<sub>n</sub>**
- **INCF <addr>,F** ;increment F -or-
- **ADDLW 0x01** ;increment W

# Multiply & Divide

## MULTIPLY

- ❖ *Unsigned* only
- ❖ First convert negative numbers (2sC) – NEG op
- ❖ Compute result sign: 0 if both signs same, 1 else (not=)
- ❖ Complement result if sign is negative – NEG op
- ❖ Other MPUs use *signed* multiply (2sC) via “Booth’s Algorithm”

## DIVIDE

- ❖ No hardware, no instruction (a few new models have a hardware divide)
- ❖ Create subroutine (may find ones in asm library)
- ❖ Compute
  - Long division
  - Non-restoring division
  - Iterative subtraction (very slow)
- ❖ Use tricks
  - Divide by **2** or any **2<sup>n</sup>**: right SHIFT by n
  - Divide by **10**: convert to BCD, then right SHIFT by 4 (reconvert to binary)
  - Divide by **5**: divide by 10, then multiply by 2 (by shifting after conv. Bin)